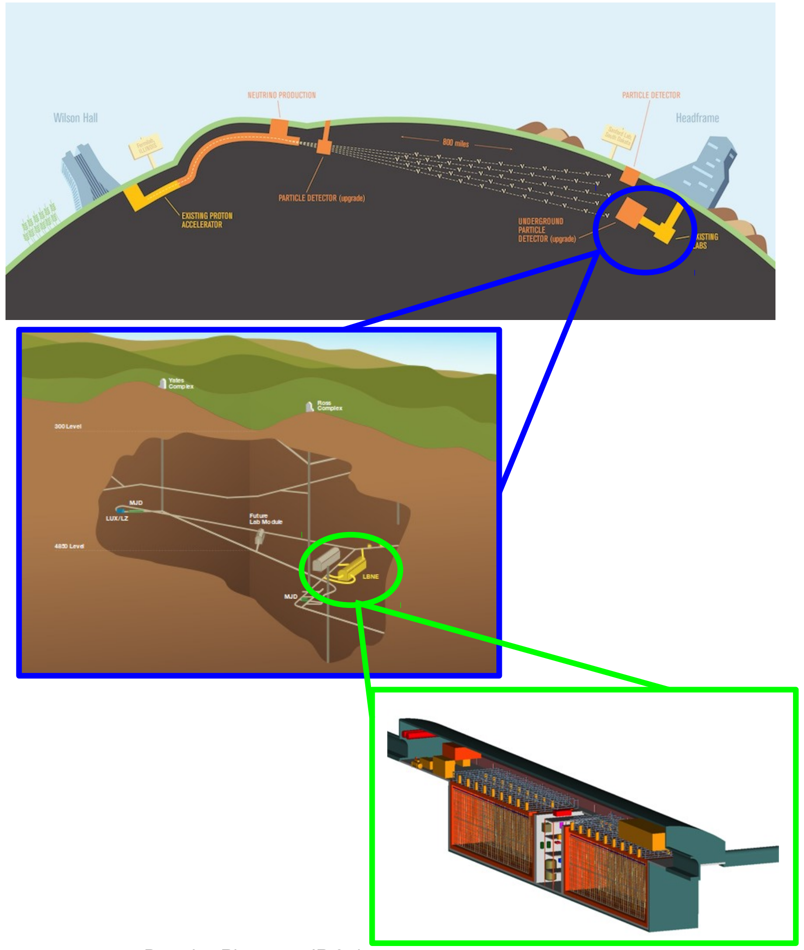
Imaging Detector Datasets

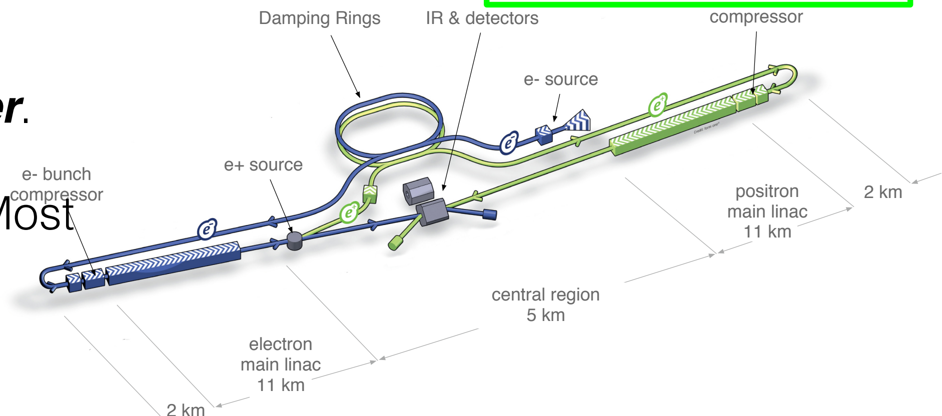
Amir Farbin

UNIVERSITY OF TEXAS ARLINGTON

Frontiers



- ***Energy Frontier***: ***Large Hadron Collider (LHC)*** at 13 TeV now, ***High Luminosity (HL)-LHC*** by 2025, perhaps 33 TeV LHC or 100 TeV Chinese machine in a couple of decades.
    - Having found Higgs, moving to studying the SM ***Higgs*** find new Higgses
    - Test ***naturalness*** (Was the Universe and accident?) by searching for New Physics like Supersymmetry that keeps Higgs light without 1 part in 10^{18} fine-tuning of parameters.
    - Find ***Dark Matter*** (reasons to think related to naturalness)
- ***Intensity Frontier***:
    - ***B Factories***: upcoming SuperKEKB/SuperBelle
    - ***Neutrino Beam Experiments***:
        - Series of current and upcoming experiments: Nova, MicroBooNE, SBND, ICURUS
        - ***US's flagship experiment*** in next decade: ***Long Baseline Neutrino Facility (LBNF)/Deep Underground Neutrino Experiment (DUNE) at Intensity Frontier***
    - Measure properties of ***b-quarks*** and ***neutrinos*** (newly discovered mass)… search for ***matter/anti-matter asymmetry***.
    - Auxiliary Physics: Study ***Supernova***. Search for ***Proton Decay*** and ***Dark Matter***.
- ***Precision Frontier***: ***International Linear Collider (ILC)***, hopefully in next decade. Most energetic $e^+ e^-$ machine.
    - ***Precision studies*** of ***Higgs*** and hopefully ***new particles*** found at LHC.

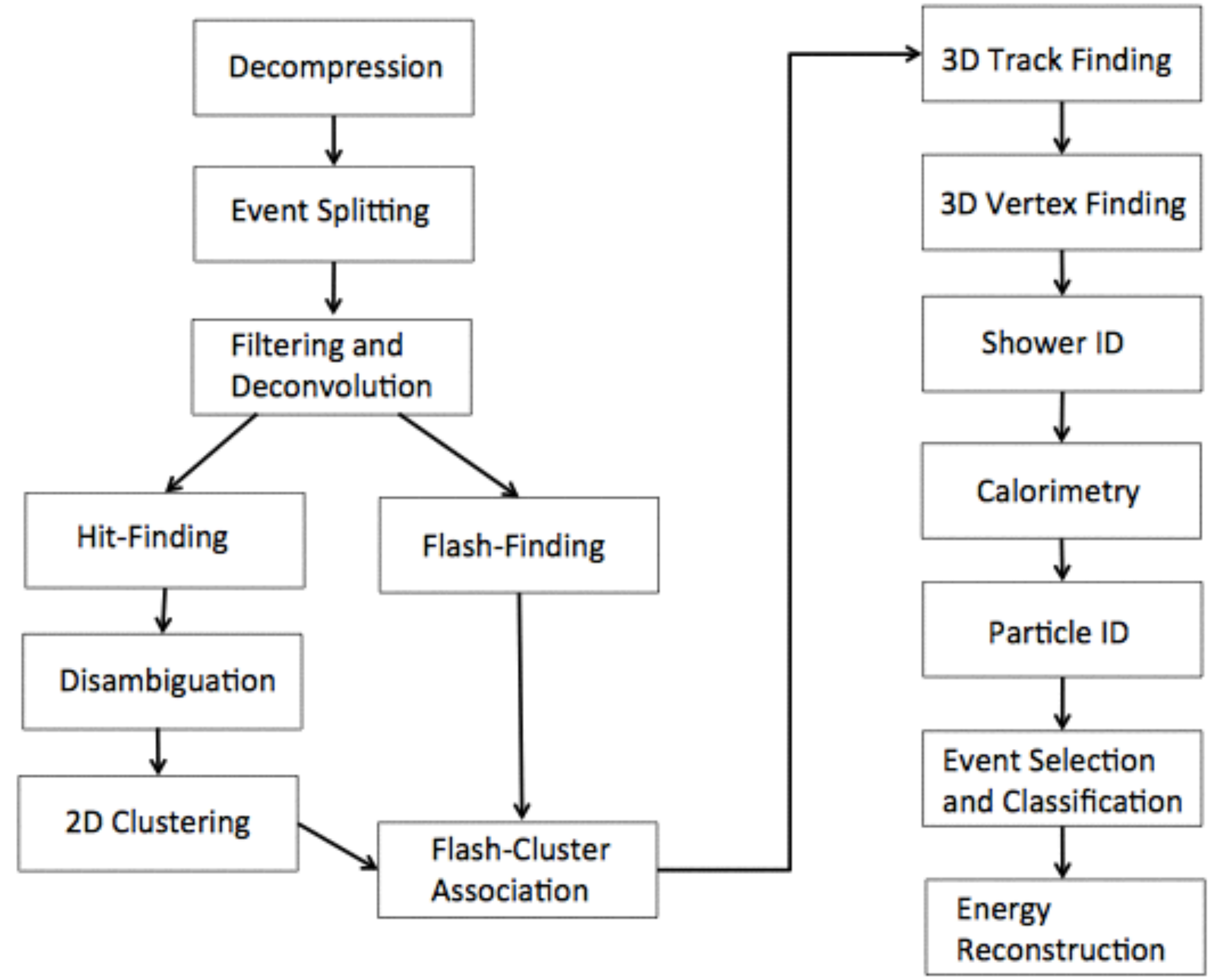


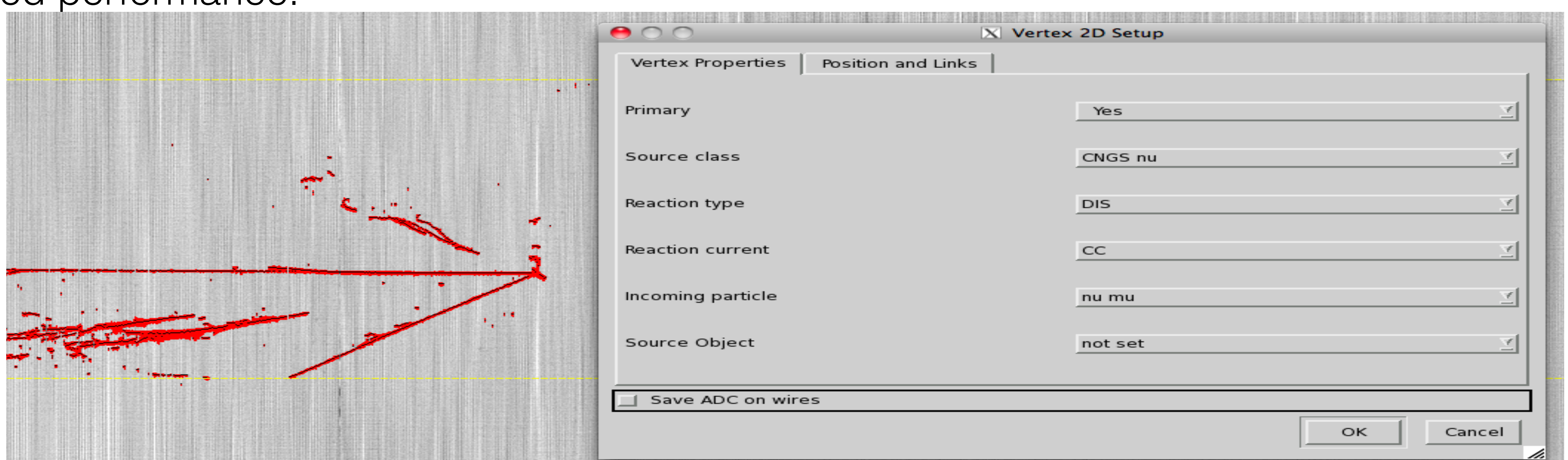
Where is ML needed?

- Traditionally ML Techniques in HEP
    - Applied to Particle/Object Identification
    - Signal/Background separation
    - Here, ML maximizes reach of existing data/detector… equivalent to additional integral luminosity.
    - There is lots of interesting work here… and potential for big impact.
- Now we hope ML can help address looming computing problems
    - Reconstruction
        - LArTPC- Algorithmic Approach very difficult
        - HL-LHC Tracking- Pattern Recognition blows up due to combinatorics
    - Simulation
        - LHC Calorimetry- Large Fraction of ATLAS CPU goes into shower simulation.

LArTPC Reco Challenge

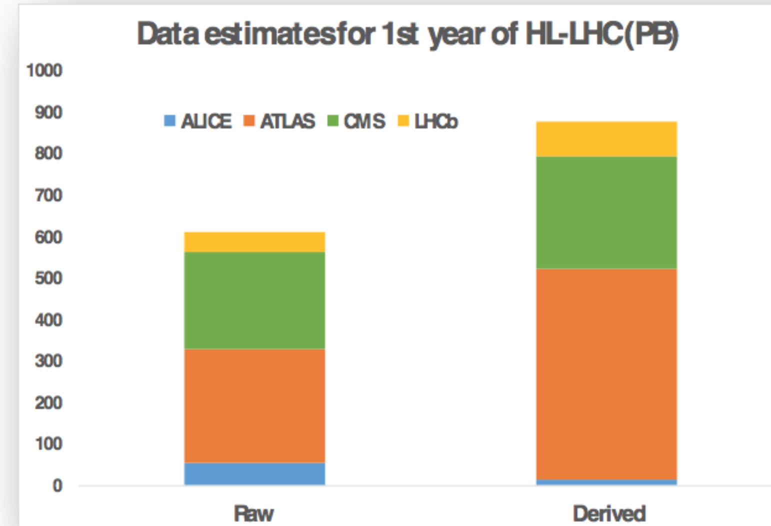
- Neutrino Physics has a long history of *hand scans*.
  - QScan: ICARUS user assisted reconstruction.
- Full automatic reconstruction has yet to be demonstrated.
  - LArSoft project:
    - art framework + LArTPC reconstruction algorithm
    - started in ArgoNeuT and contributed to/used by many experiments.
  - Full neutrino reconstruction is still far from expected performance.



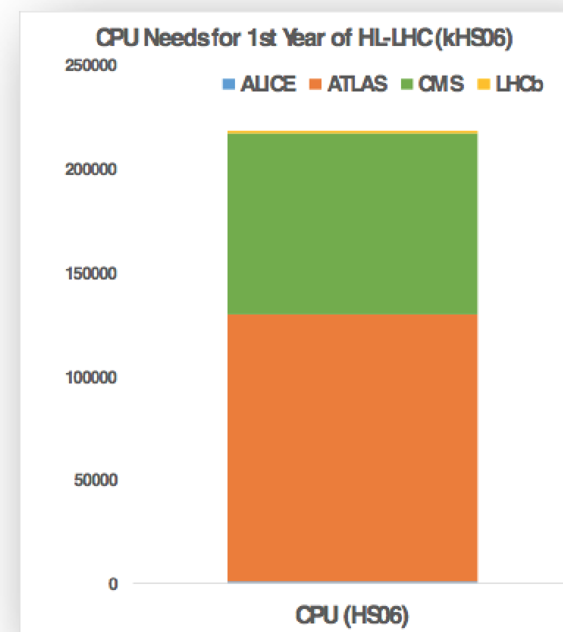


Computing Challenge

- ***Computing*** is perhaps the biggest challenge for the HL-LHC
  - ***Higher Granularity*** = larger events.
  - ***O(200) proton collision / crossing: tracking pattern recognition combinatorics becomes untenable***.
  - O(100) times data = multi ***exabyte datasets***.
  - ***Moore's law has stalled***: Cost of adding more transistors/silicon area no longer decreasing…. for processors. Many-core co-processors still ok.
    - Naively we need 60x more CPU, with 20%/year Moore's law giving only 6-10x in 10-11 years.
  - Preliminary estimates of ***HL-LHC computing budget many times larger than LHC***.
- ***Solutions***:
  - ***Leverage opportunistic resources and HPC*** (most computation power in highly parallel processors).
  - ***Highly parallel processors*** (e.g. GPUs) are already > 10x CPUs for certain computations.
    - Trend is away from x86 towards ***specialized hardware*** (e.g. GPUs, Mics, FPGAs, Custom DL Chips)
    - Unfortunately parallelization (i.e. Multi-core/GPU) has been extremely difficult for HEP.



Data:
- Raw 2016: 50 PB → 2027: 600 PB
- Derived (1 copy): 2016: 80 PB → 2027: 900 PB



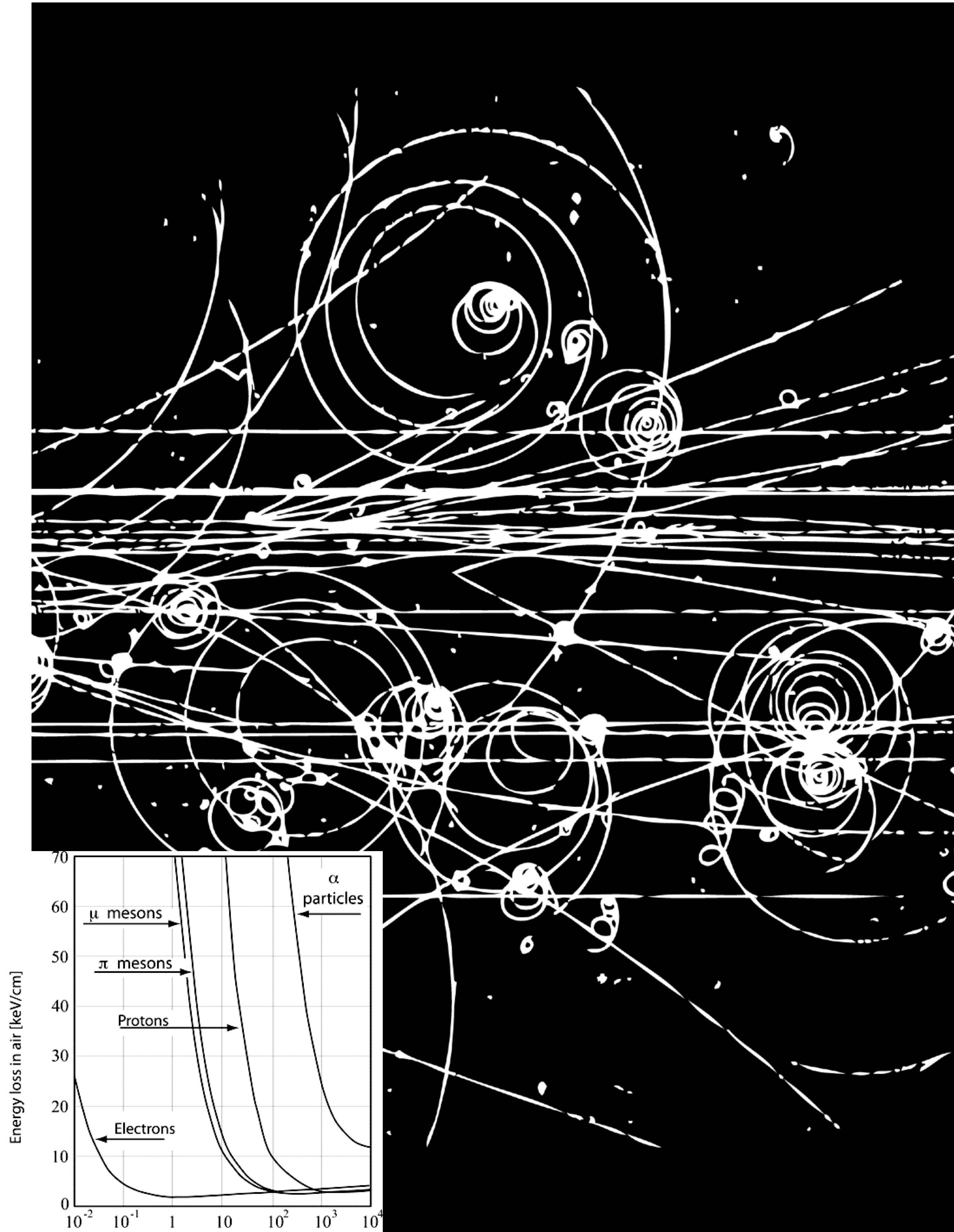
CPU:
- x60 from 2016

From WLCG Workshop Intro, Ian Bird, 8 Oct, 2016

Reconstruction

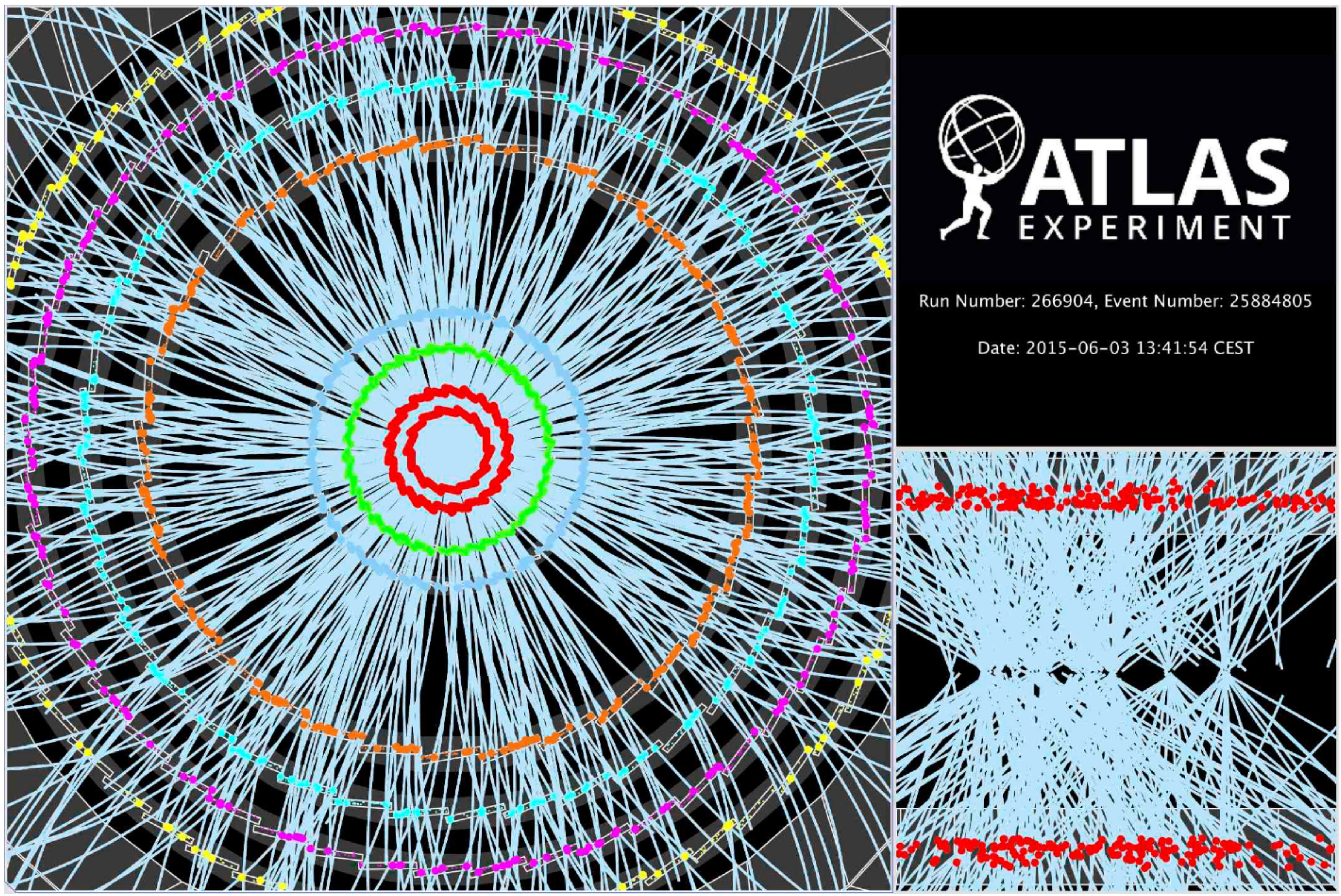
How do we “see” particles?

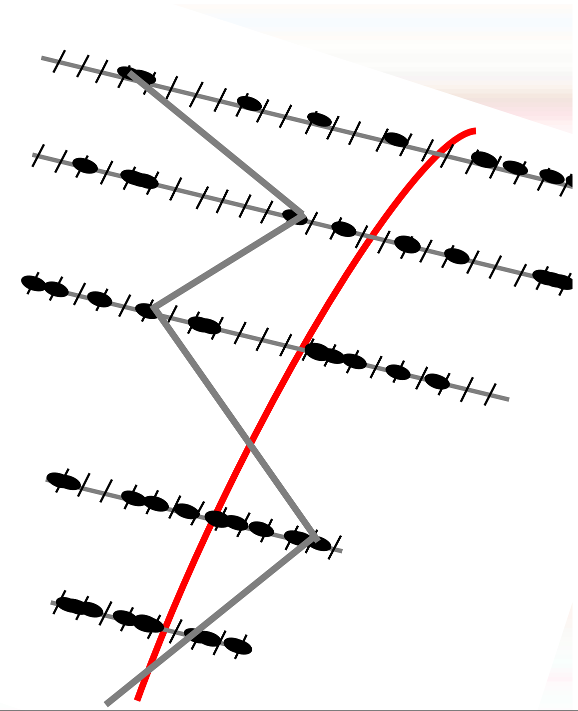
- ***Charged particles ionize media***
  - Image the ions.
    - In ***Magnetic Field*** the ***curvature*** of trajectory ***measures momentum***.
    - Momentum resolution degrades as less curvature: $\sigma(p) \sim c\, p \oplus d$.
      - d due to multiple scattering.
  - Measure ***Energy Loss*** (~ # ions)
    - dE/dx = Energy Loss / Unit Length = f(m, v) = Bethe-Block Function
      - Identify the particle type
    - ***Stochastic process*** (Laudau)
  - Loose all energy → range out.
    - Range characteristic of particle type.

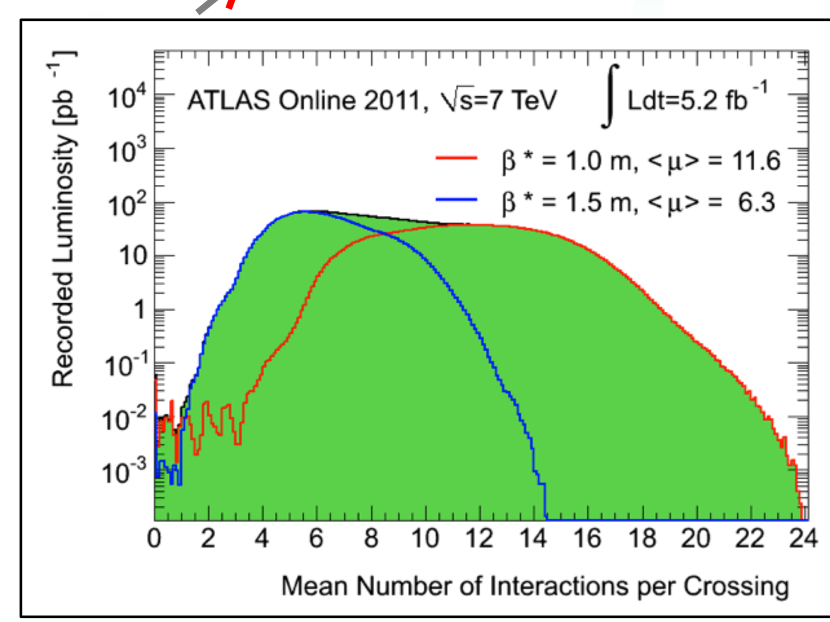


Tracking

- Measure Charged particle trajectories. If B-field, then measure momentum.

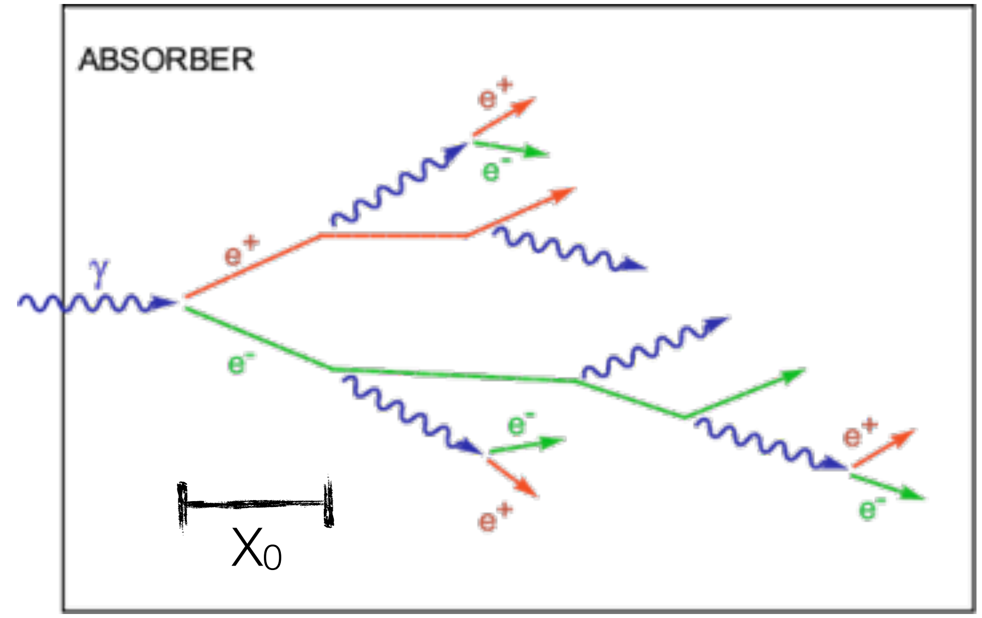


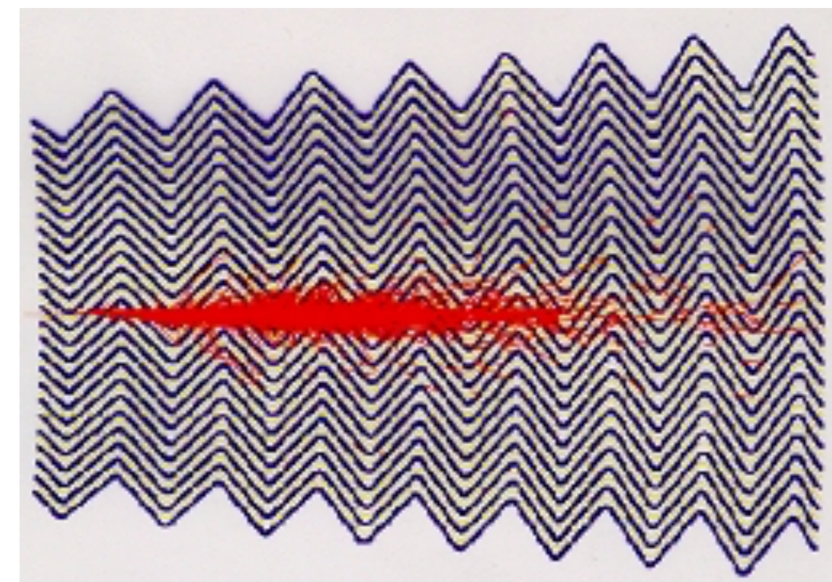


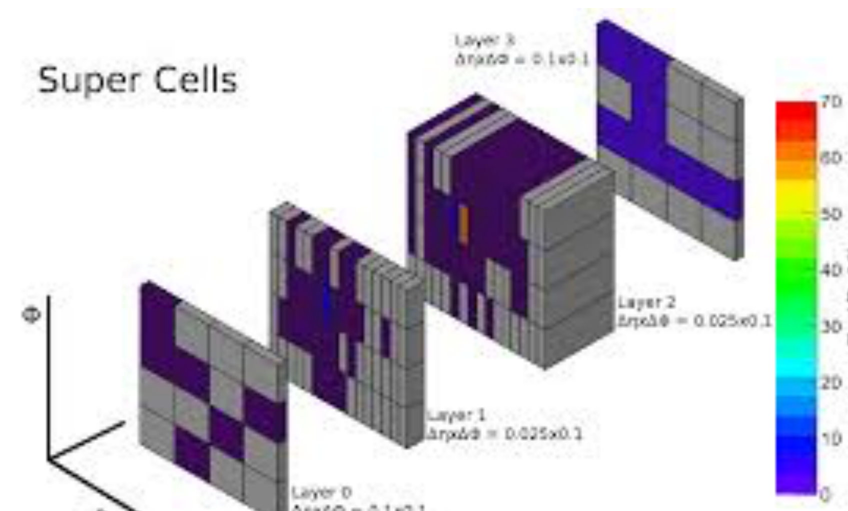


How do we “see” particles?

- Particles deposit their energy in a ***stochastic process*** know as ***“showering”***, secondary particles, that in turn also shower.
    - Number of secondary particles ~ Energy of initial particle.
    - Energy resolution improves with energy: $\sigma(E) / E = a/\sqrt{E} \oplus b/E \oplus c$.
        - a = sampling, b = noise, c = leakage.
    - Density and Shape of shower characteristic of type of particle.
- ***Electromagnetic calorimeter***: Low Z medium
    - ***Light particles***: electrons, photons, $\pi^0 \rightarrow \gamma\gamma$ interact with electrons in medium
- ***Hadronic calorimeters***: High Z medium
    - ***Heavy particles***: Hadrons (particles with quarks, e.g. charged pions/protons, neutrons, or jets of such particles)
        - Punch through low Z.
        - Produce secondaries through strong interactions with the nucleus in medium.
        - Unlike EM interactions, not all energy is observed.

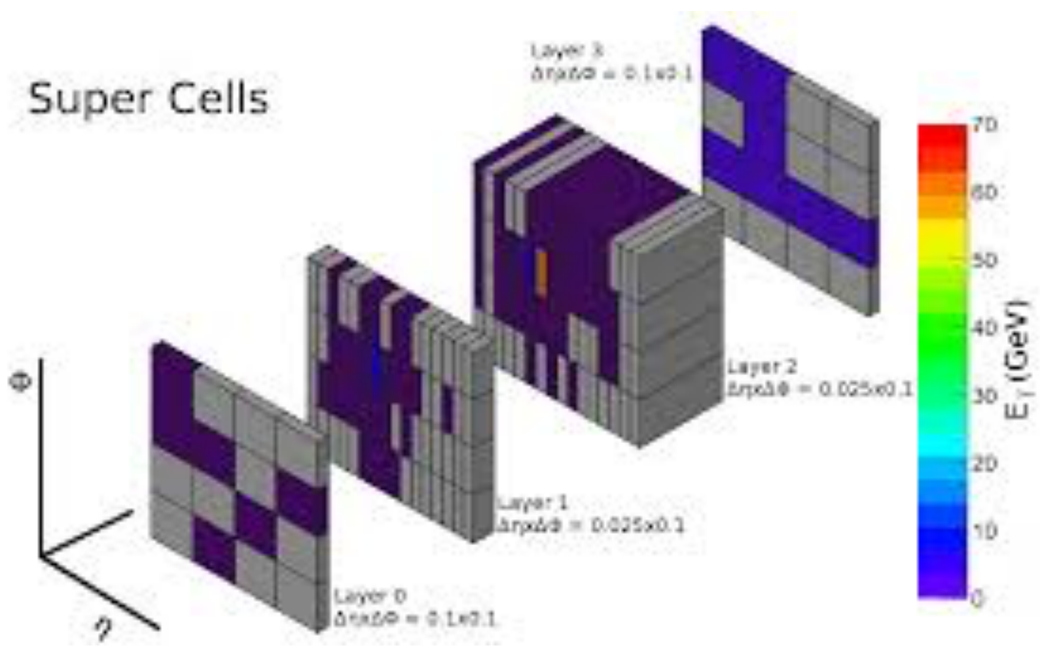


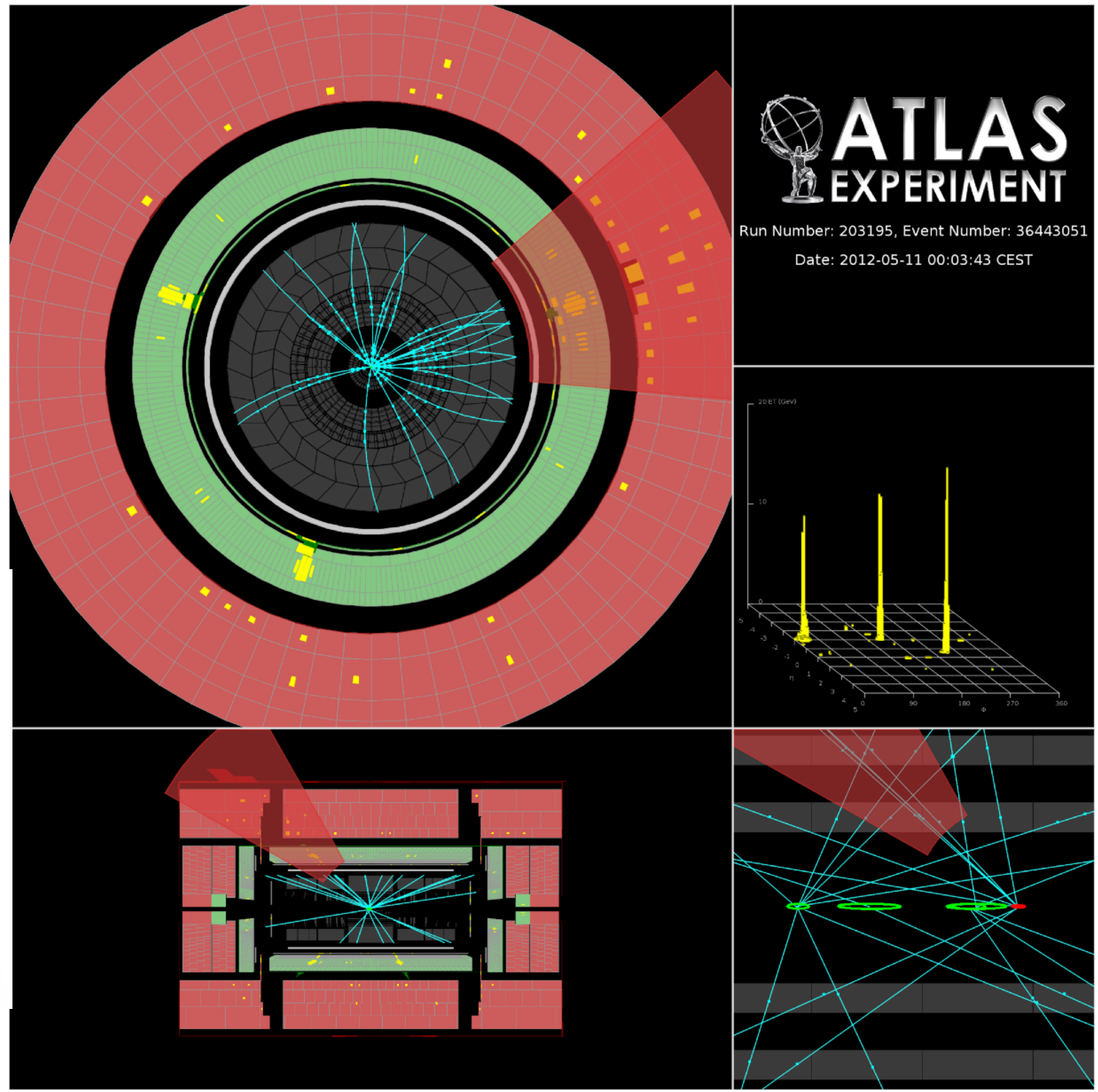




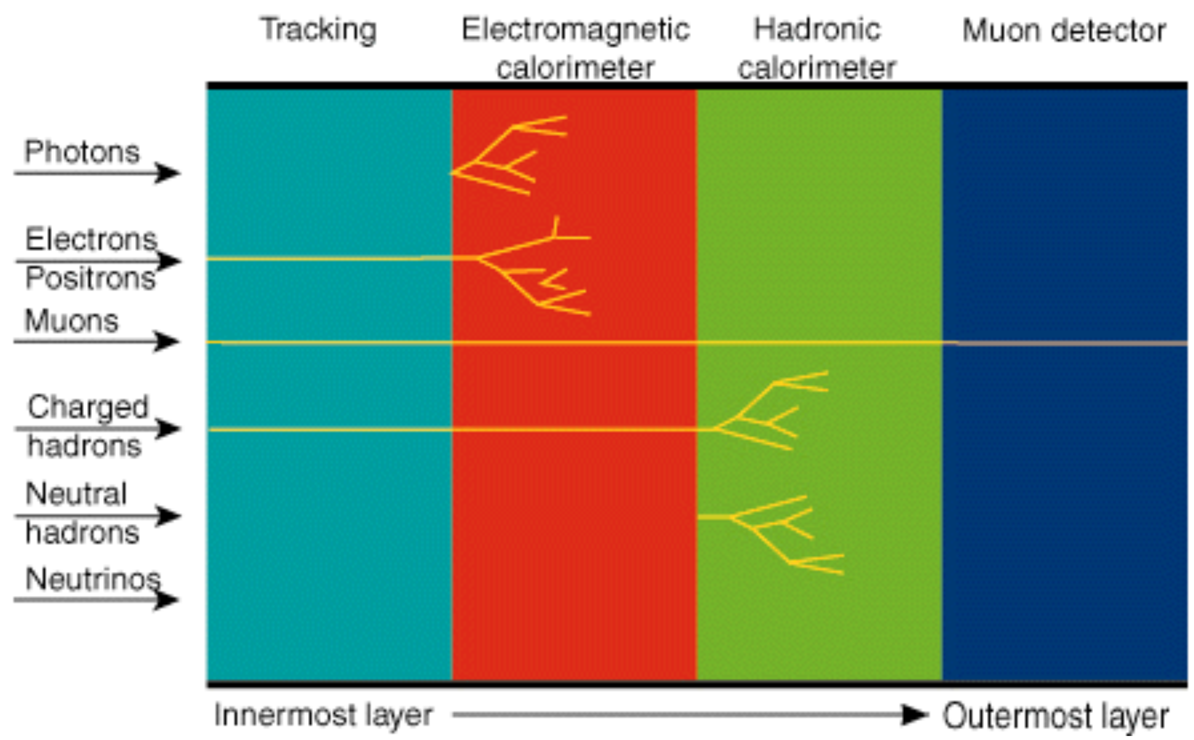
Calorimetry

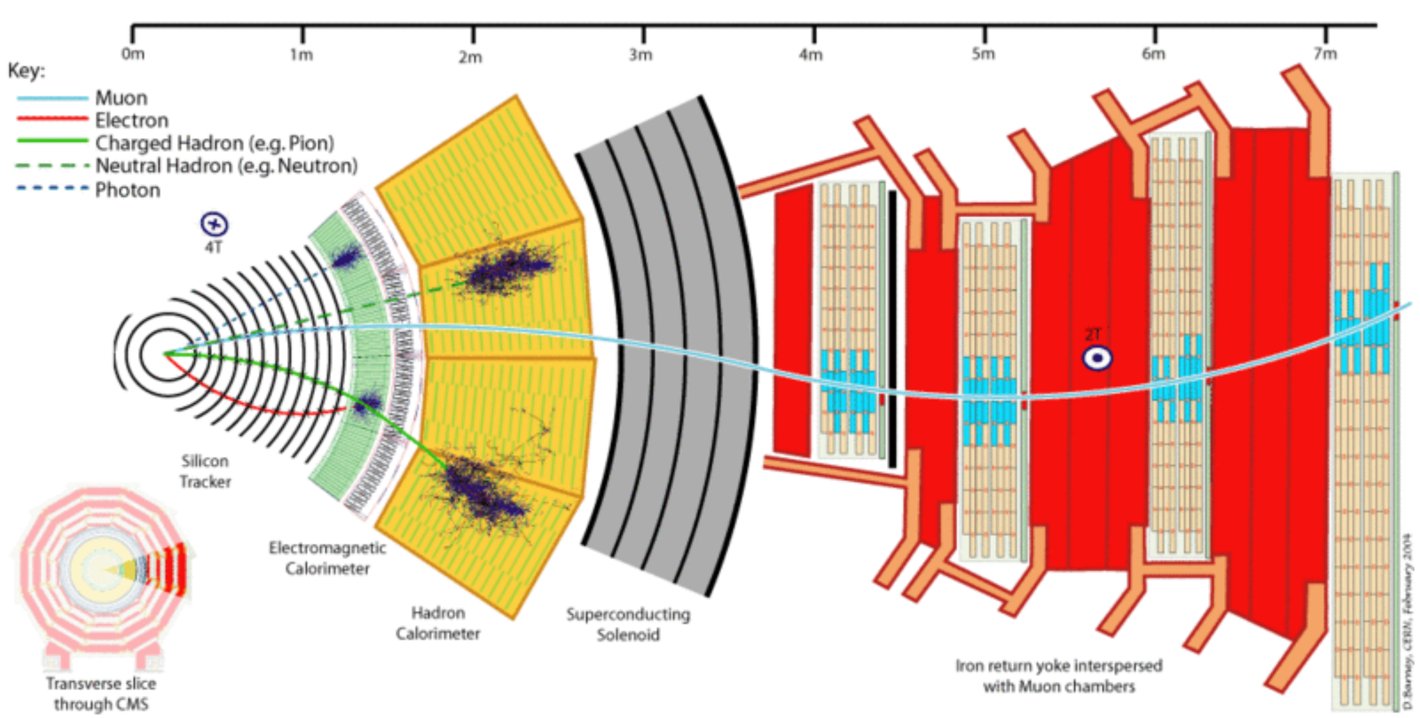
- Make particle interact and loose all energy, which we measure. 2 types:
  - Electromagnetic: e.g. crystals in CMS, Liquid Argon in ATLAS.
  - Hadronic: e.g. steel + scintillators
  - e.g ATLAS:
    - 200K Calorimeter cells measure energy deposits.
    - 64 x 36 x 7 3D Image

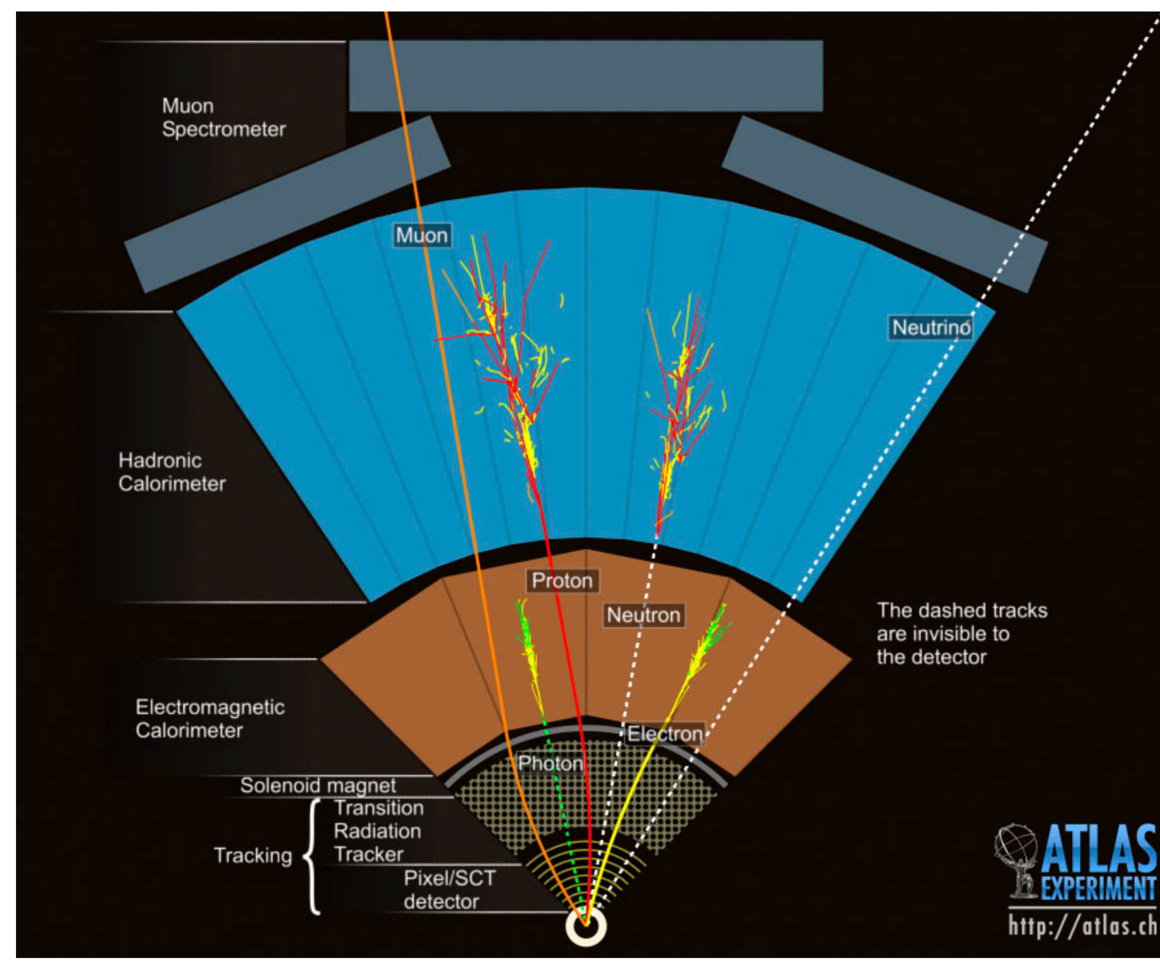




LHC/ILC detectors

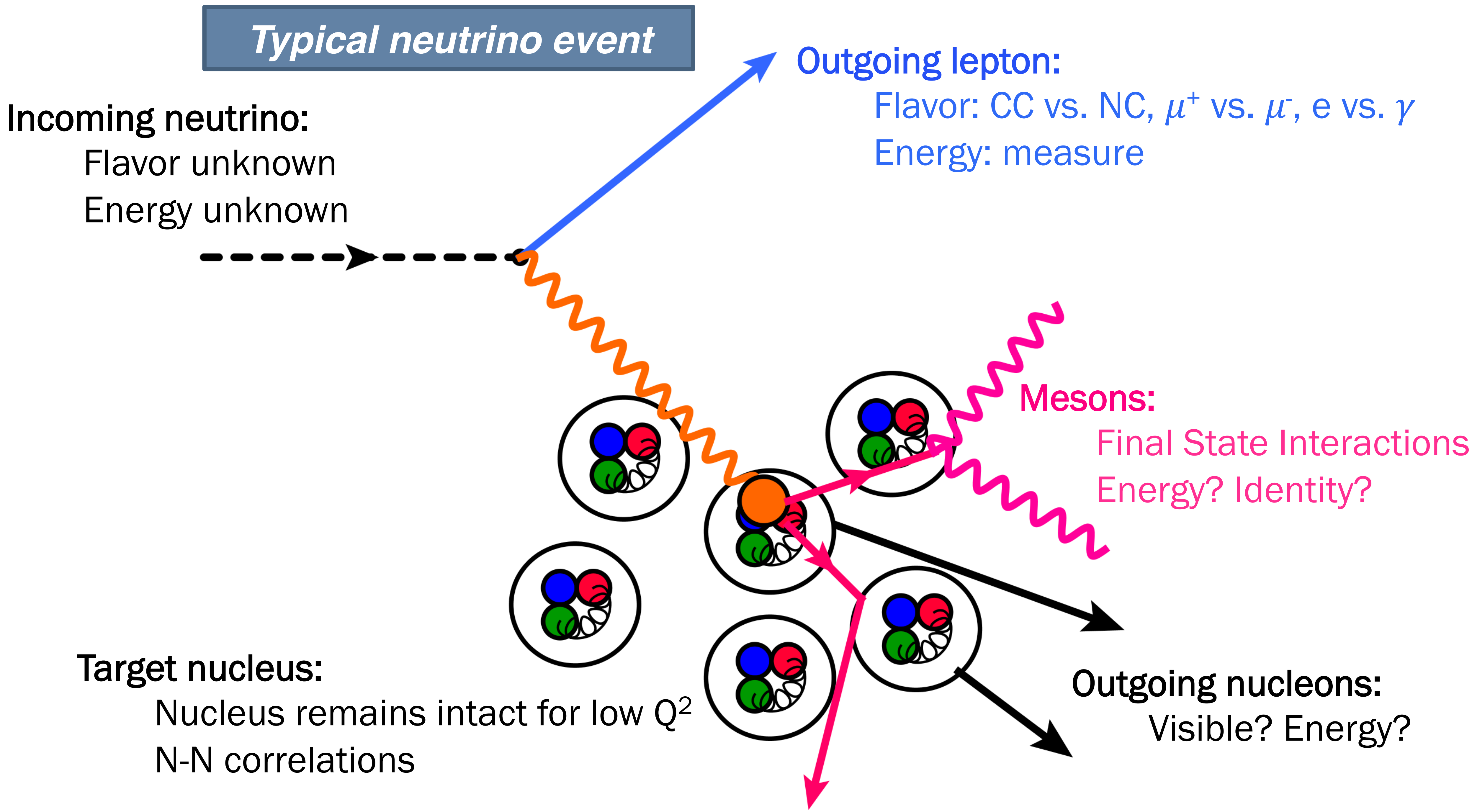






Neutrino Detection

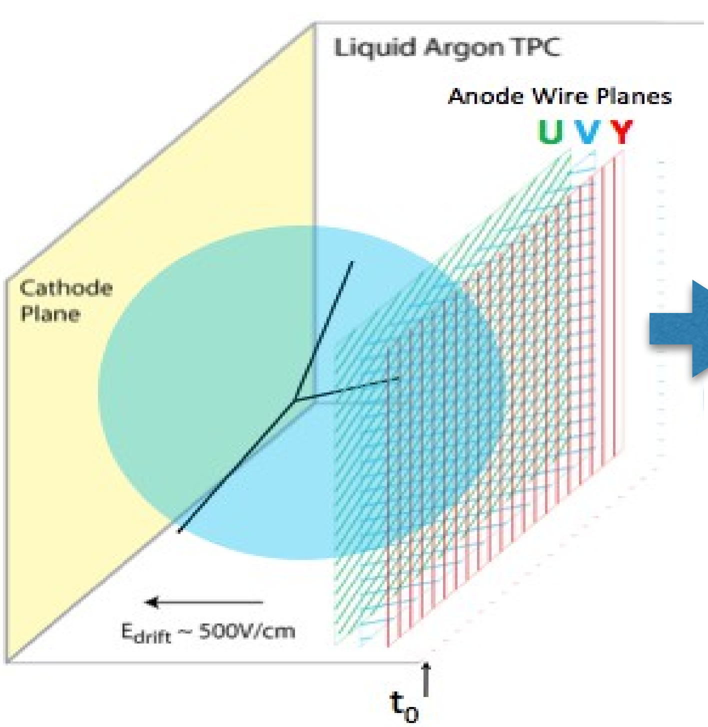
In neutrino experiments, try to determine flavor and estimate energy of incoming neutrino by looking at outgoing products of the interaction.

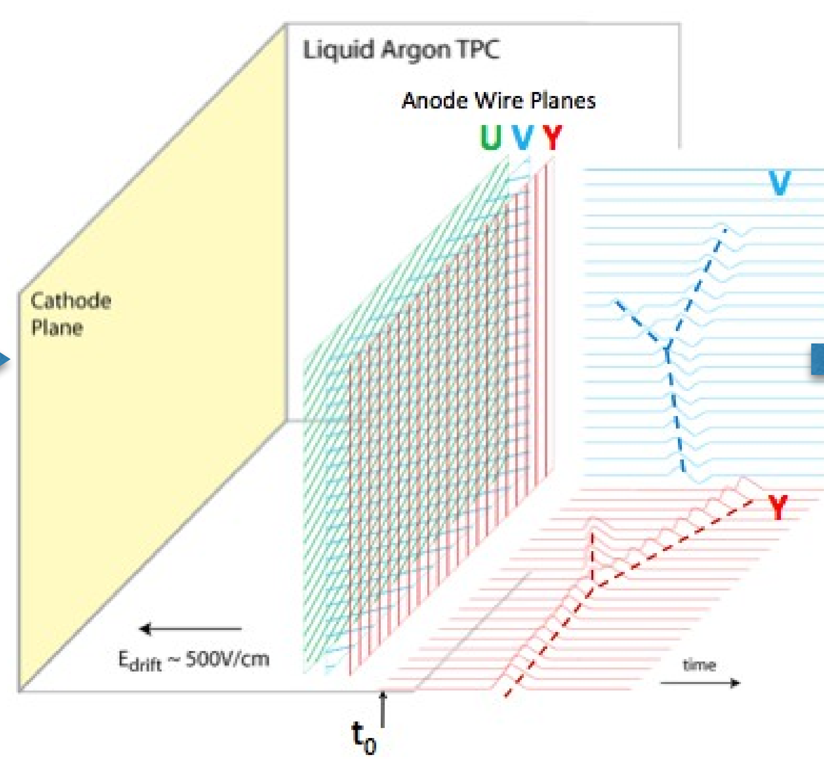


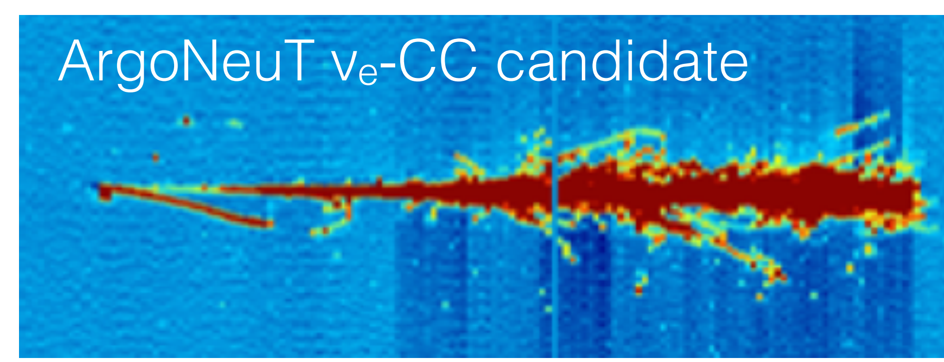
Jen Raaf

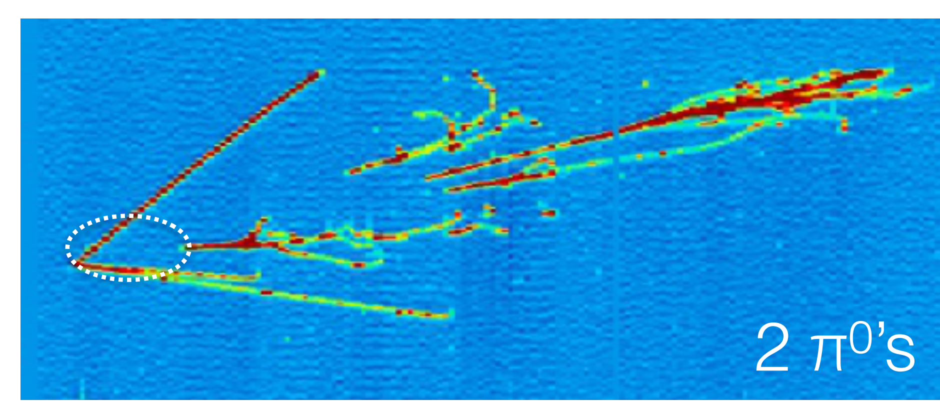
Neutrino Detectors

- ***Need large mass/volume*** to maximize chance of neutrino interaction.
- Technologies:
  - Water/Oil Cherenkov
  - Segmented Scintillators
  - ***Liquid Argon Time Projection Chamber: promises ~ 2x detection efficiency.***
    - ***Provides tracking, calorimetry, and ID all in same detector***.
    - Chosen technology for US's flagship LBNF/DUNE program.
    - Usually 2D read-out… 3D inferred.
  - Gas TPC: full 3D

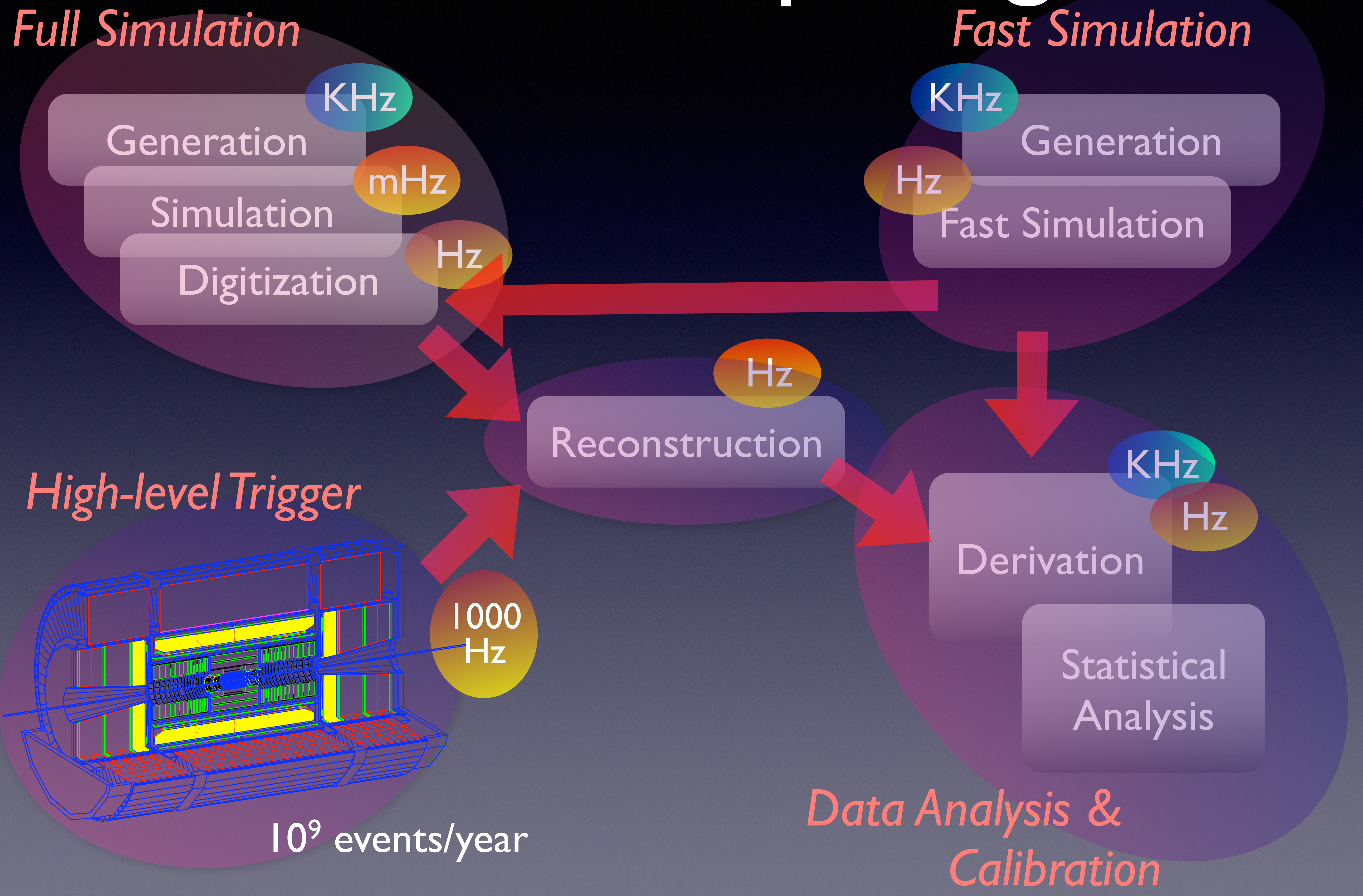






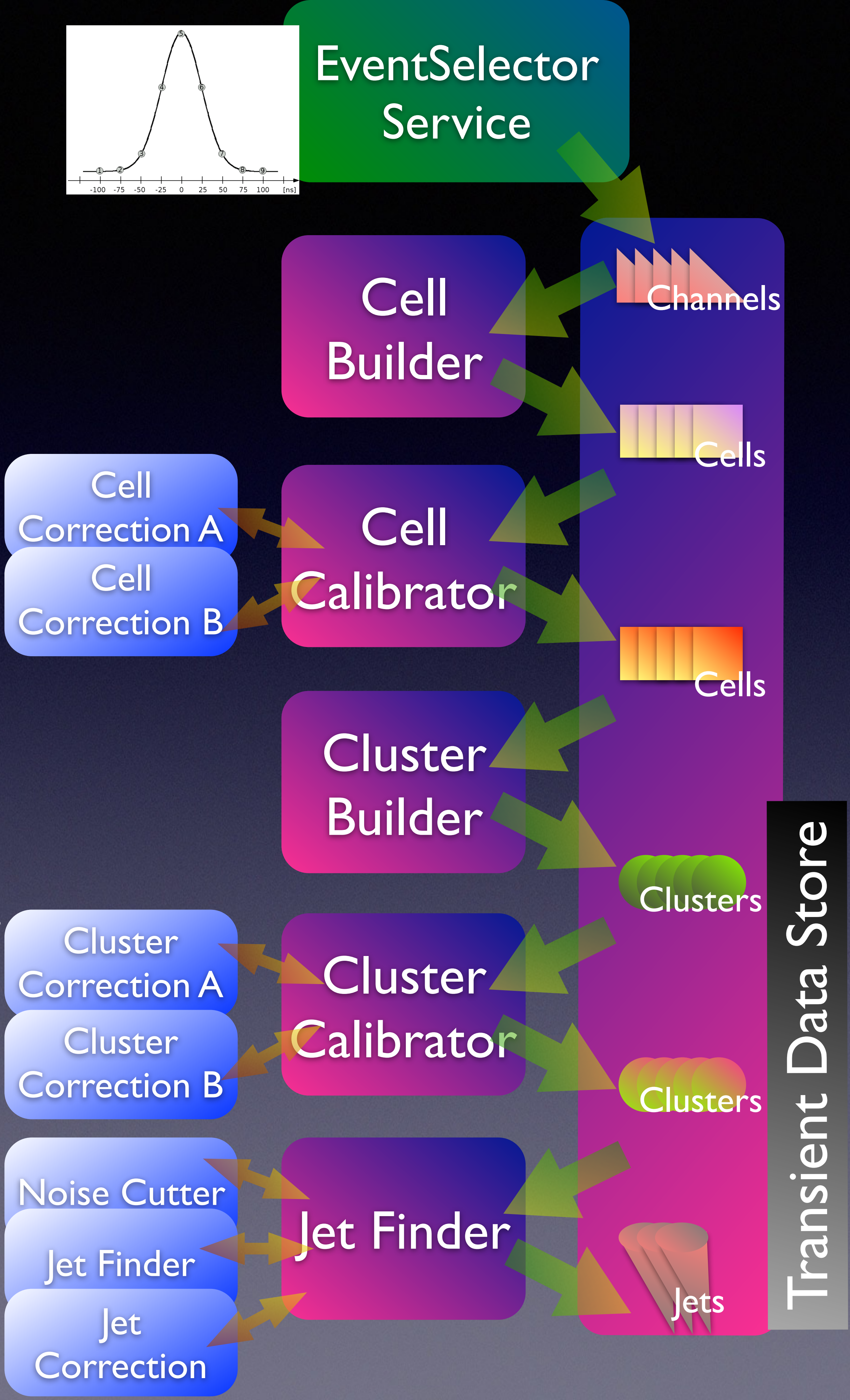


HEP Computing



Reconstruction

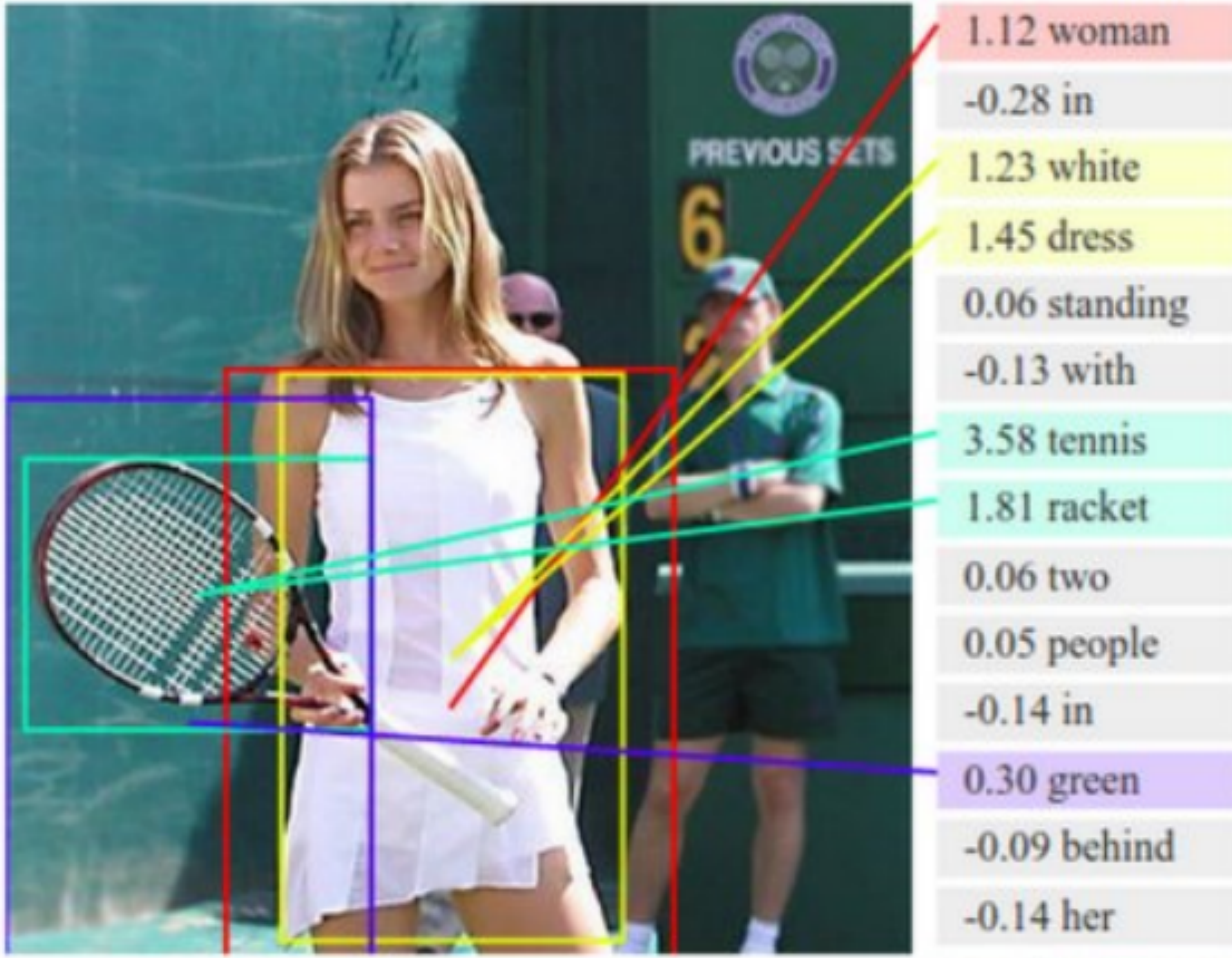
- Starts with ***raw inputs*** (e.g. Voltages)
- Low level ***Feature Extraction***: e,g, Energy/Time in each Calo Cell
- ***Pattern Recognition***: Cluster adjacent cells. Find hit pattern.
- ***Fitting***: Fit tracks to hits.
- ***Combined reco***: e.g.:
  - Matching Track+EM Cluster = Electron.
  - Matching Track in inter detector + muon system = Muon
- ***Output particle candidates*** and measurements of their properties (e.g. energy)

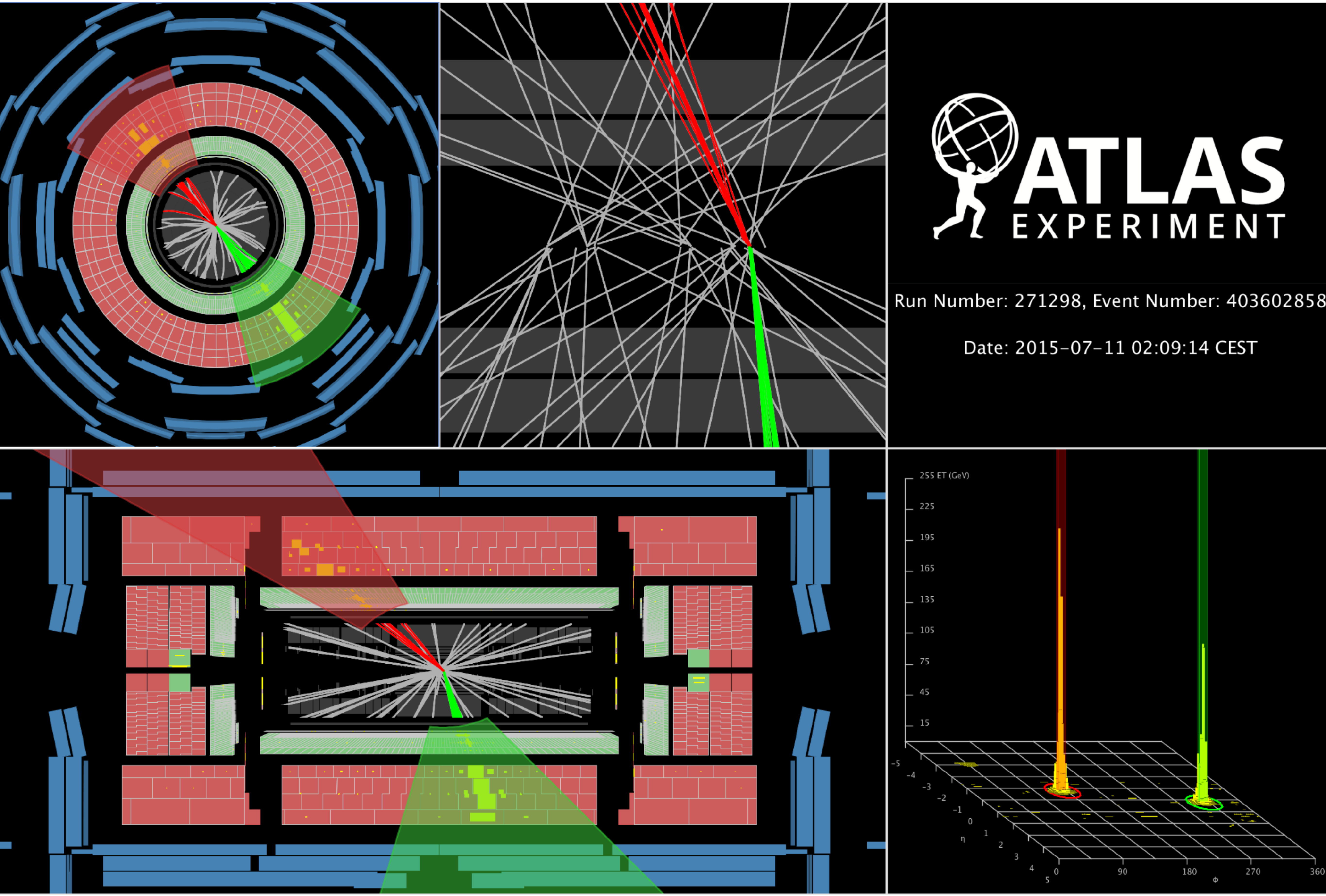


Deep Learning

Why go Deep?

- ***Better Algorithms***
  - DNN-based classification/regression generally **out perform** hand crafted algorithms.
  - In some cases, it may provide a ***solution*** where ***algorithm approach doesn't exist or fails***.
  - ***Unsupervised learning***: make sense of complicated data that we don't understand or expect.
- ***Easier Algorithm Development***: ***Feature Learning*** instead of *Feature Engineering*
  - Reduce time physicists spend writing developing algorithms, ***saving time and cost***. (e.g. ATLAS > $250M spent software)
  - Quickly perform performance ***optimization*** or ***systematic studies***.
- ***Faster Algorithms***
  - After training, DNN inference is often *faster* than sophisticated algorithmic approach.
  - DNN can ***encapsulate expensive computations***, e.g. Matrix Element Method.
  - ***Generative Models*** enable fast simulations.
  - ***Already parallelized*** and optimized for GPUs/HPCs.
  - ***Neuromorphic*** processors.


PREVIOUS SETS
6
1.12 woman
-0.28 in
1.23 white
1.45 dress
0.06 standing
-0.13 with
3.58 tennis
1.81 racket
0.06 two
0.05 people
-0.14 in
0.30 green
-0.09 behind
-0.14 her


ATLAS
EXPERIMENT
Run Number: 271298, Event Number: 403602858
Date: 2015-07-11 02:09:14 CEST
255 ET (GeV)
225
195
165
135
105
75
45
15
-5
-4
-3
-2
-1
0
1
2
3
4
5
η
0
90
180
270
360
Φ

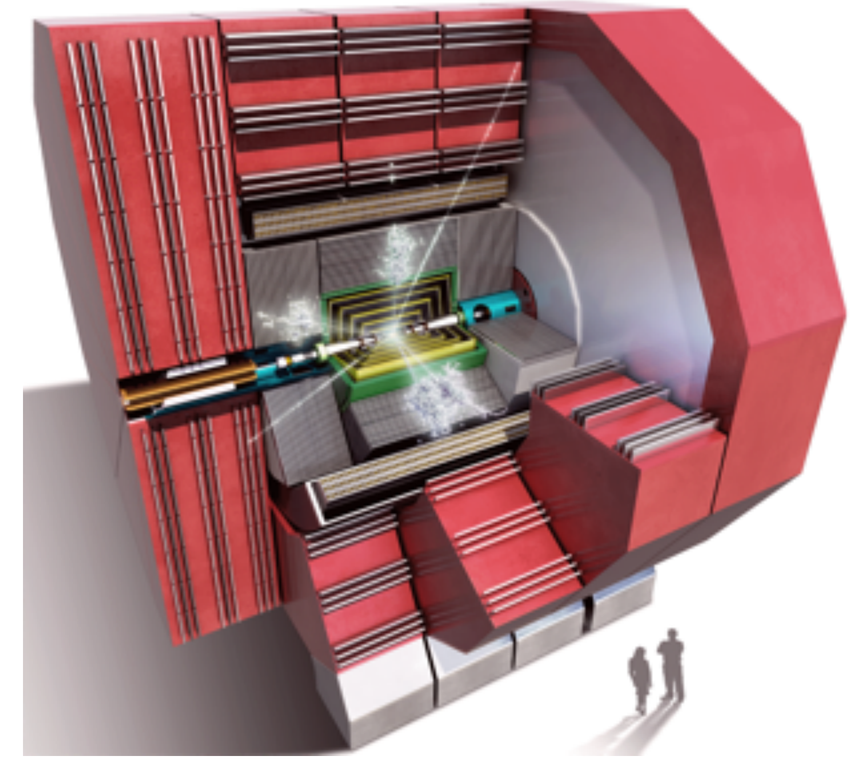
Datasets

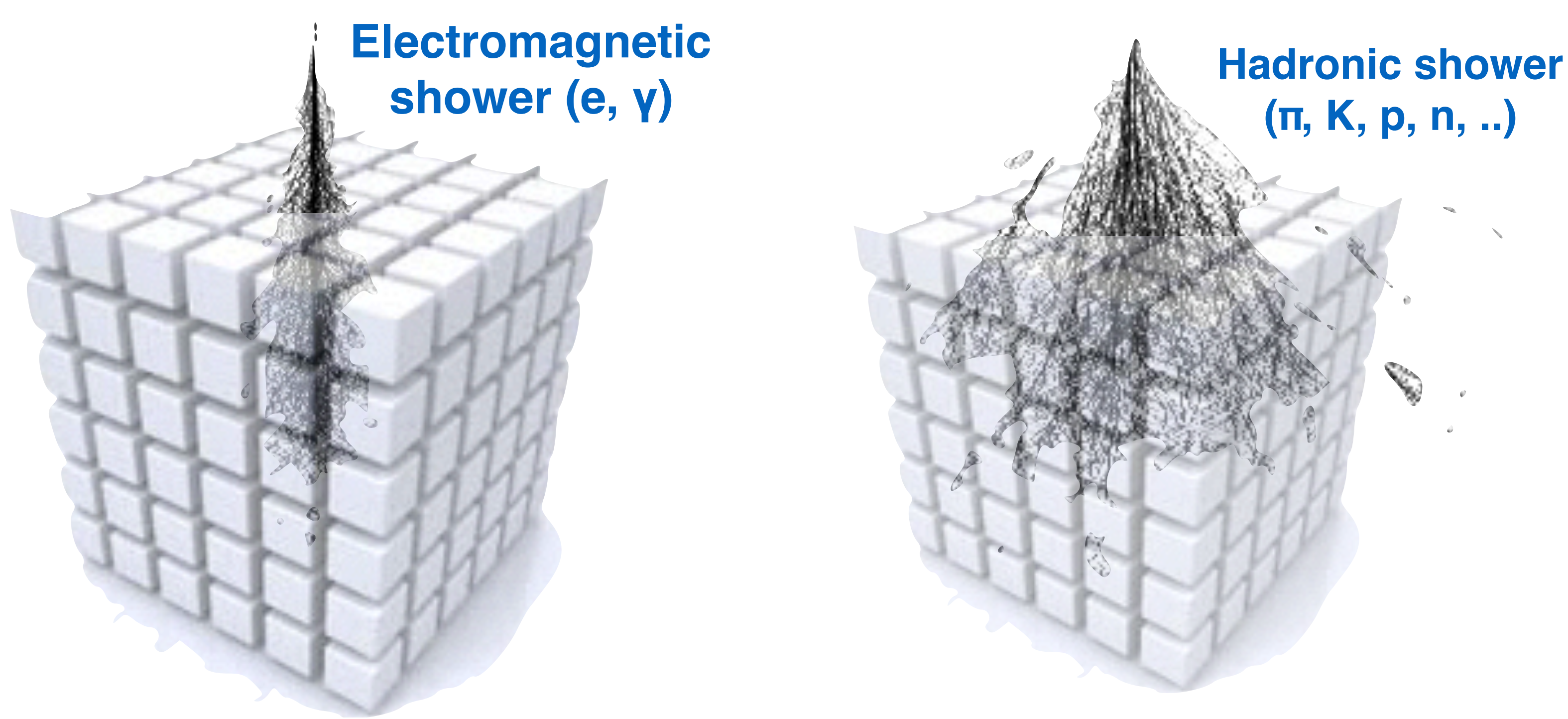
Public Datasets

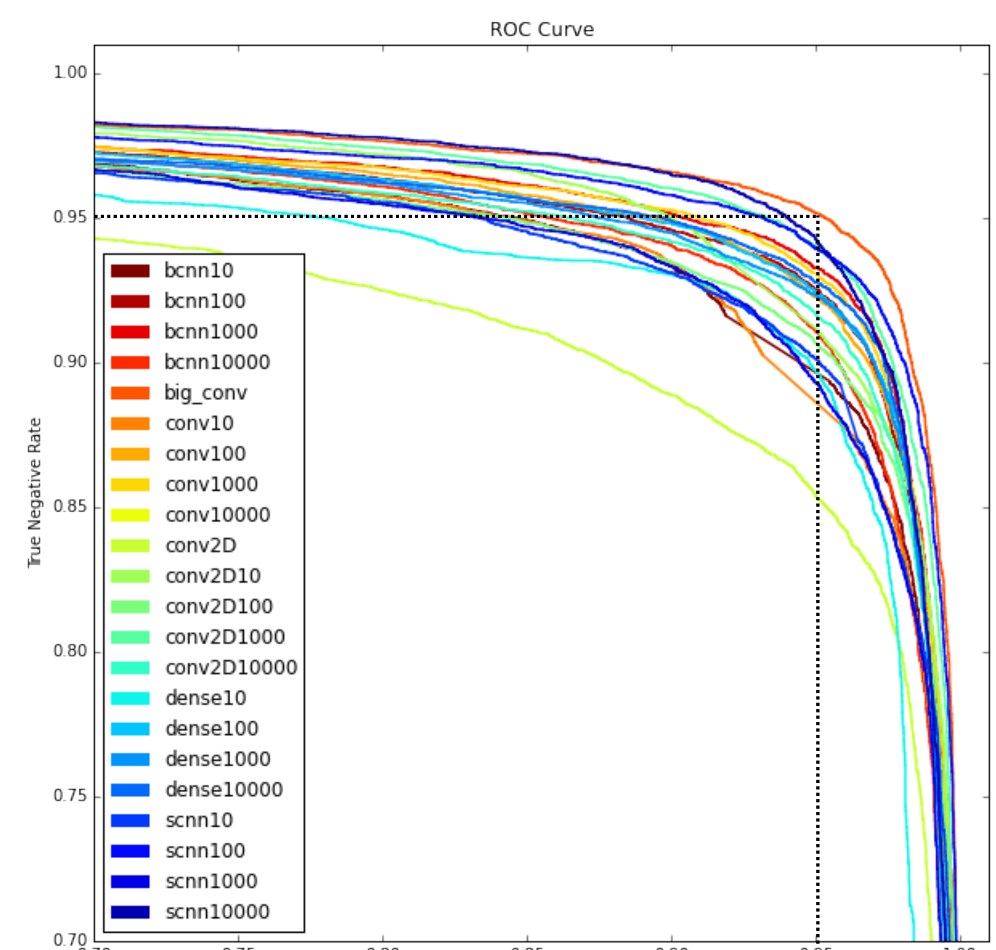
- ***Biggest obstacles*** to DNN research is ***Data accessibility***.
  - Detector level studies require ***CPU intensive simulations***.
  - DNNs require large training sets with ***full level of detail*** (i.e. not 4-vectors).
  - Experiments have such samples, but they are not easily accessible and ***not public***.
  - ***Difficult to collaborate*** with DL community or other experiments.
- ***Public datasets:***
  - We provide data, tools (e.g. fast data read), fully setup problems. Goal is build working groups around each dataset.
  - ***LArTPC*** (Sepideh Shahsavarani, AF): LArIAT detector. 1 M of every particle species (including neutrinos).
    - Challenges: Particle/Neutrino Classification and Energy Reco, Noise Suppression, 2D->3D.
  - ***Calorimetry*** (Maurizio Pierini, Jean-Roch Vlimant, Nikita Smirnov, AF): LCD Calorimeter.
    - Challenges: PID/Energy Reco. Simulation.
  - ***Tracking***
    - Simple 2D tracking data shown at Connecting the Dots will be used for DS@HEP.
    - *TrackingML/ACTS* (David Rousseau, Andreas Salzberger, …) HL-LHC like detector/environment.
  - ***CMS Jets***: Full Reco Simulated Jets for boosted object and jet ID

Calorimeter Dataset

- CLIC is a proposed CERN project for a linear accelerator of electrons and positrons to TeV energies (~ LHC for protons)
    - LCD is a detector concept.
    - Not a real experiment yet, so we could simulate data and make it public.
- The LCD calorimeter is an array of absorber material and silicon sensors comprising the most granular calorimeter design available
    - Data is essentially a 3D image

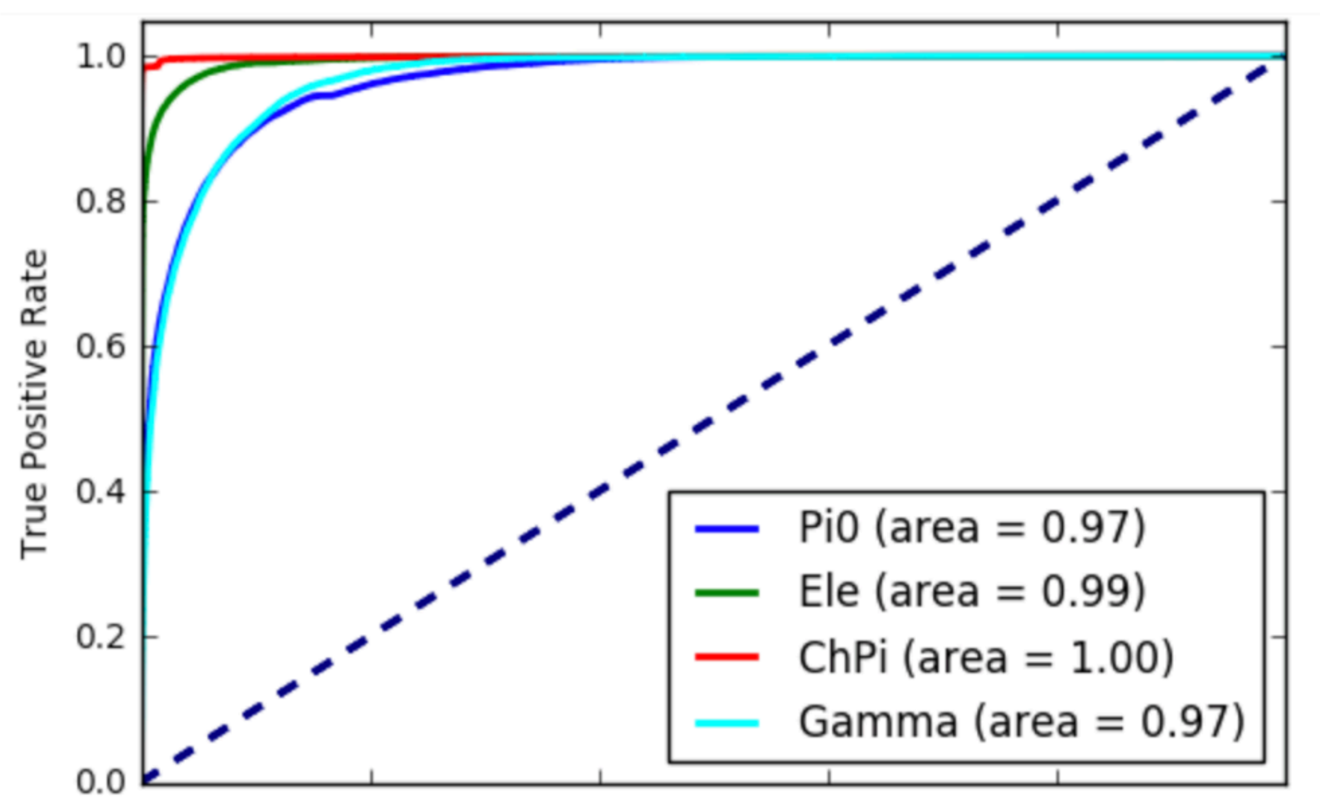


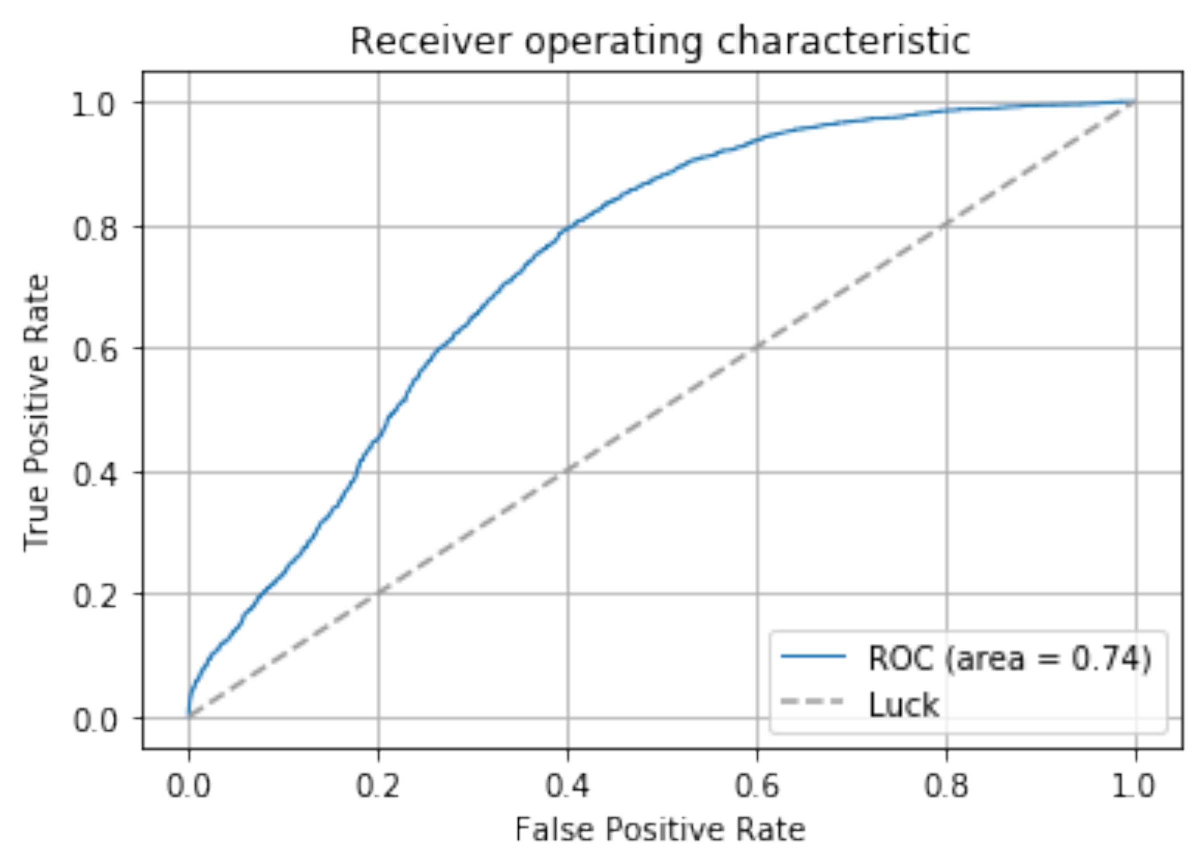




DNN vs BDT

- The classification problem, as setup, ends up being very simple.
  - The real backgrounds are jets, not single particles.
  - V2 of dataset will address this shortcoming
- Comparison to BDT trained on features





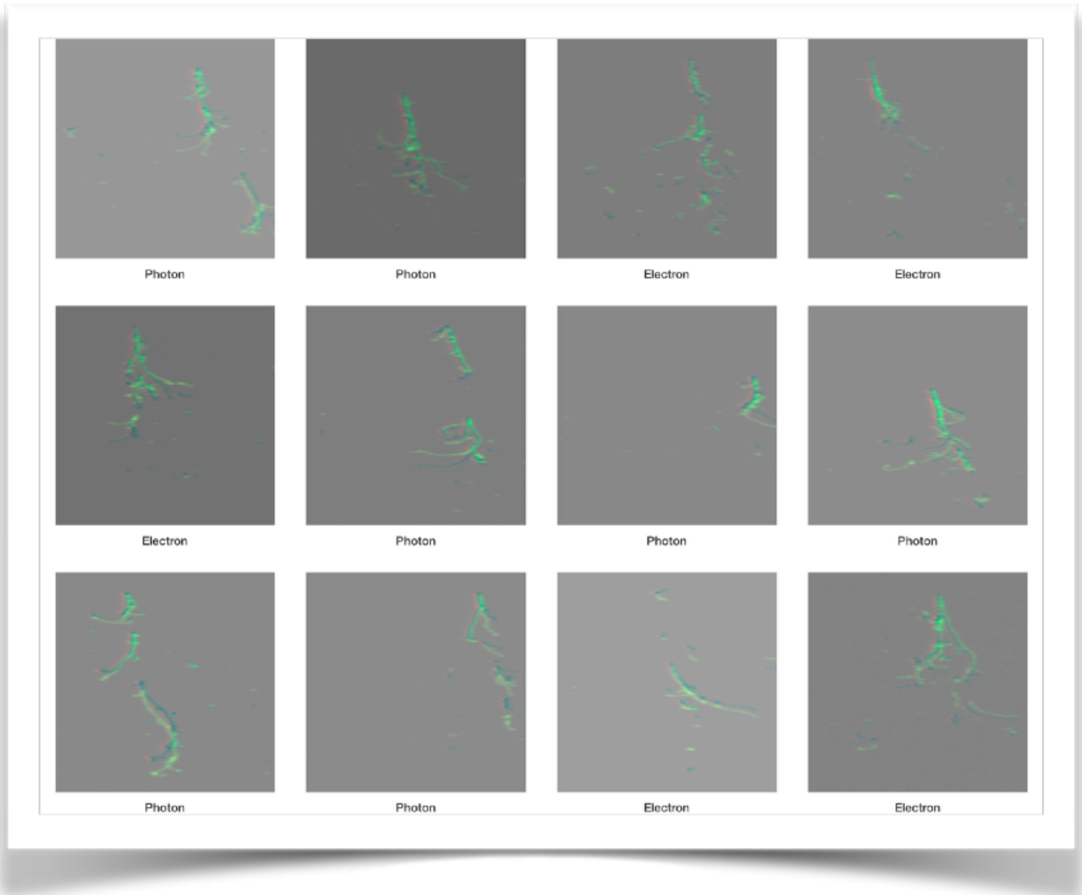
LCD Data Details

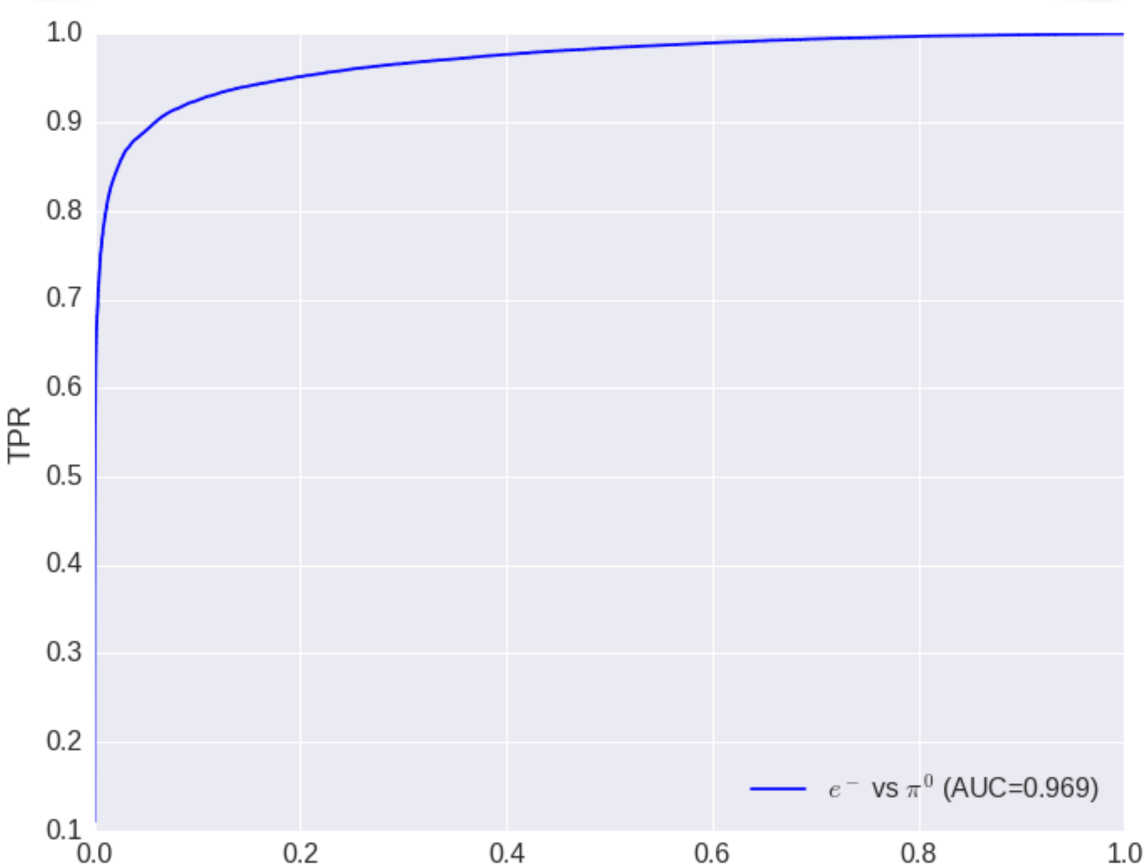
- 4 particle types, separate into directories. Needs to be mixed for training.
- Images:
  - ECAL: 25x25x25 cell section of calorimeter around particle.
  - HCAL: 5x5x60 cell section of calorimeter around particle.
- True Energy and PDG ID
- Features:
  - 'ECALMeasuredEnergy', 'ECALNumberOfHits', 'ECAL_ratioFirstLayerToTotalE', 'ECAL_ratioFirstLayerToSecondLayerE', 'ECALMoment1X', 'ECALMoment2X', 'ECALMoment3X', 'ECALMoment4X', 'ECALMoment5X', 'ECALMoment6X', 'ECALMoment1Y', 'ECALMoment2Y', 'ECALMoment3Y', 'ECALMoment4Y', 'ECALMoment5Y', 'ECALMoment6Y', 'ECALMoment1Z', 'ECALMoment2Z', 'ECALMoment3Z', 'ECALMoment4Z', 'ECALMoment5Z', 'ECALMoment6Z', 'ECAL_HCAL_ERatio', 'ECAL_HCAL_nHitsRatio'

LCD Dataset Challenges/ Tasks

1. Classification
    - With existing setup, get excellent performance with simple DNN (not a CNN).
2. Energy Regression (Wednesday)
    - Hasn't been looked at…
    - Interesting issues, e.g. accounting for known calorimetric resolution.
3. Generative Models (Wednesday)
    - One of the primary challenges.

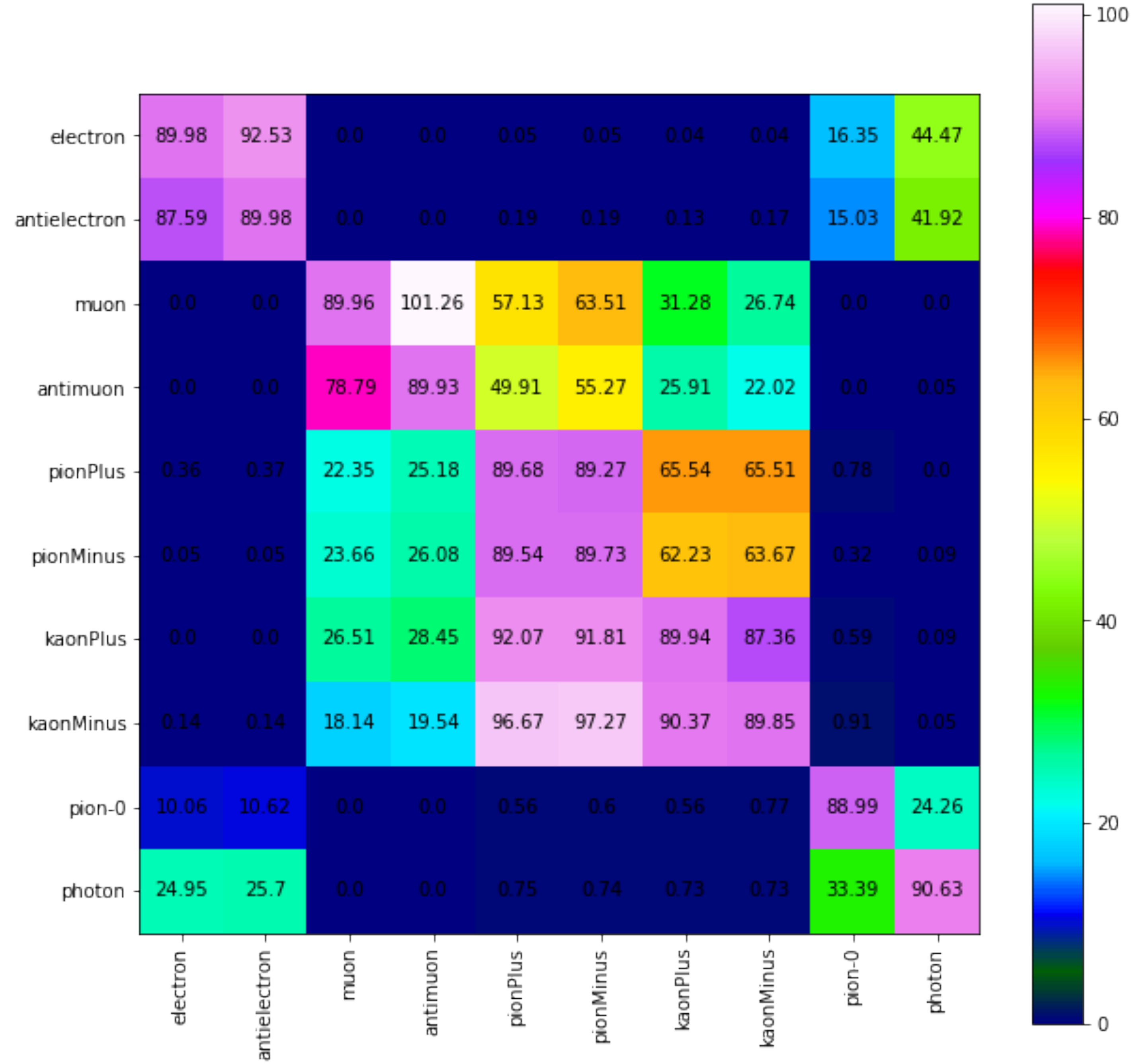
LArTPC Dataset





- Training samples have been at best ~100k examples…. usually much less.
- My students (S. Shahsavarani and G. Hilliard) simulated a huge sample of LArTPC events (LArIAT Detector).
  - Necessitated by Energy Regression studies.
  - 1 M of every particle species: $e^{\pm}$, $p^{\pm}$, $K^{\pm}$, $\pi^{\pm}$, π^{0}, $\mu^{\pm}$, γ, ν_e, ν_μ, ν_τ
  - Flat Energy distribution.
- Note that though this data is large, LArIAT is the smallest LArTPC detector with 2 x 240 wires.
  - DUNE will have 1 M wires.
- Have been working with P. Sadowski (UCI) to build inception-based CNN.

LArIAT: DNN vs Alg

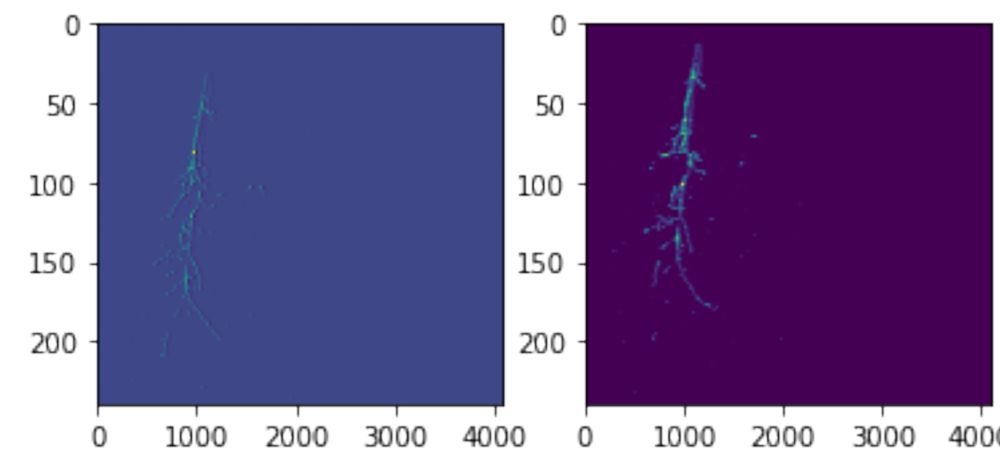


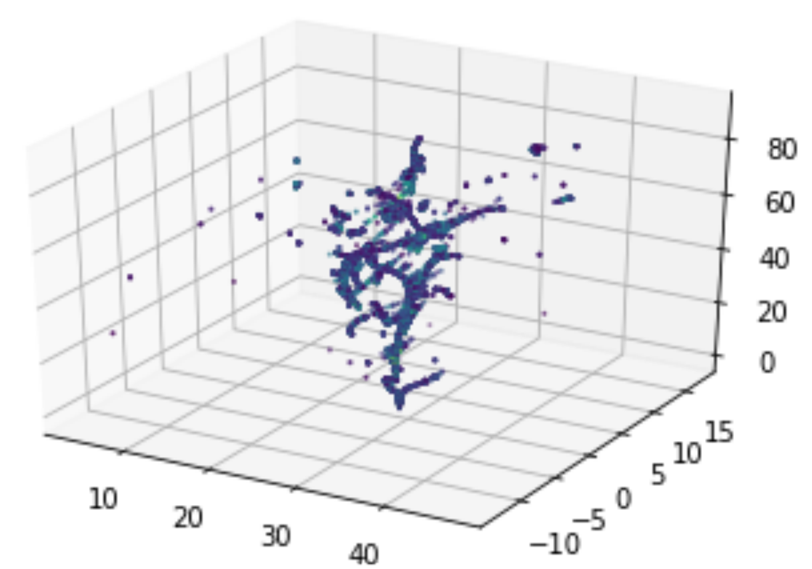
	π+	κ+	μ+	e+	γ
DNN	74.42%	40.67%	6.37%	0.12%	0%
LArIAT	74.5%	68.8%	88.4%	6.8%	2.4%

	π−	κ−	μ−	e−	γ
DNN	78.68%	54.47%	13.54%	0.11%	0.25%
LArIAT	78.7%	73.4%	91.0%	7.5%	2.4%

LArTPC Data Details

- 1 M of each particle type. Separate files for each files for each particle type.
  - For training they need to be mixed.
  - Images are large, so they are usually down-sampled.
  - Subset today… about 2.2 TB.
- Each “event” is two types of files:
  - 2D: LArTPC Reconstruction + True Info
    - images: (NEvents, 2, 240, 4096)
    - True: Energy, Px, Py, Pz,
    - Neutrino Truth: lep_mom_truth, nu_energy_truth, mode_truth
    - Track_length
  - 3D: Truth only
    - trajectory/C: x,y,z of charge deposits
    - trajectory/V: deposited charge

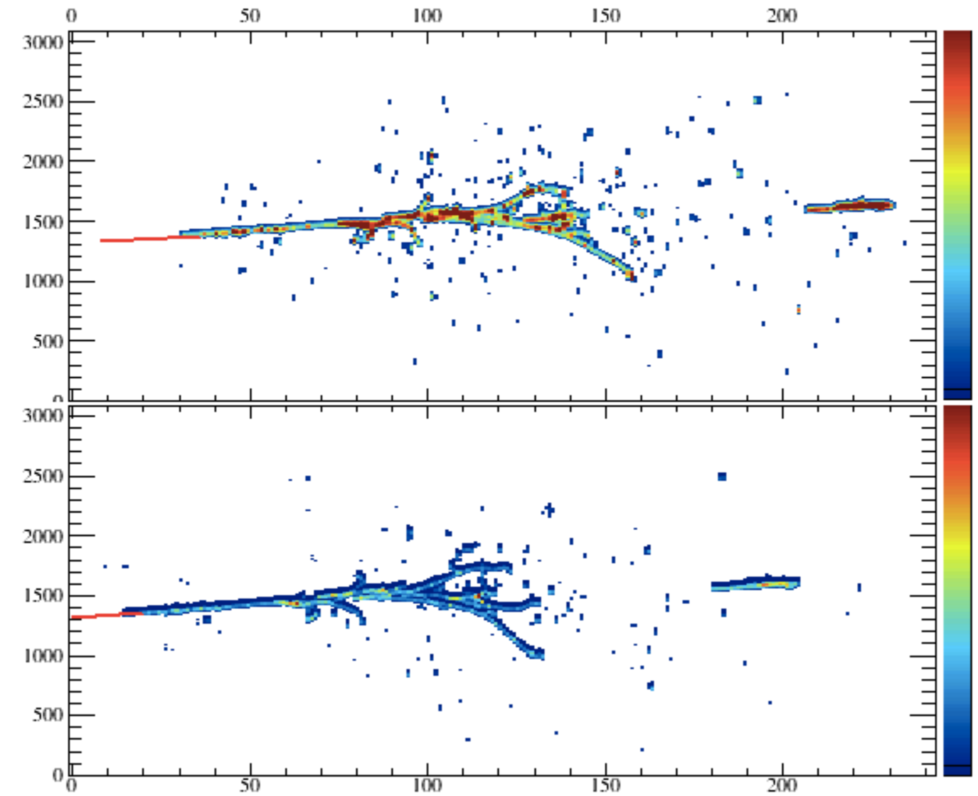




LArTPC Challenges/Tasks

1. Classification: (Monday)

- Automatic reconstruction has proven to be very challenging
- CNNs have shown to perform better on classification… on down sampled data.
- Neither has achieved the performance assumed to be achievable for DUNE to achieve
    - Particles: ~90% efficiency, 1% fake
    - Neutrino: ~80% efficiency, 1% fake

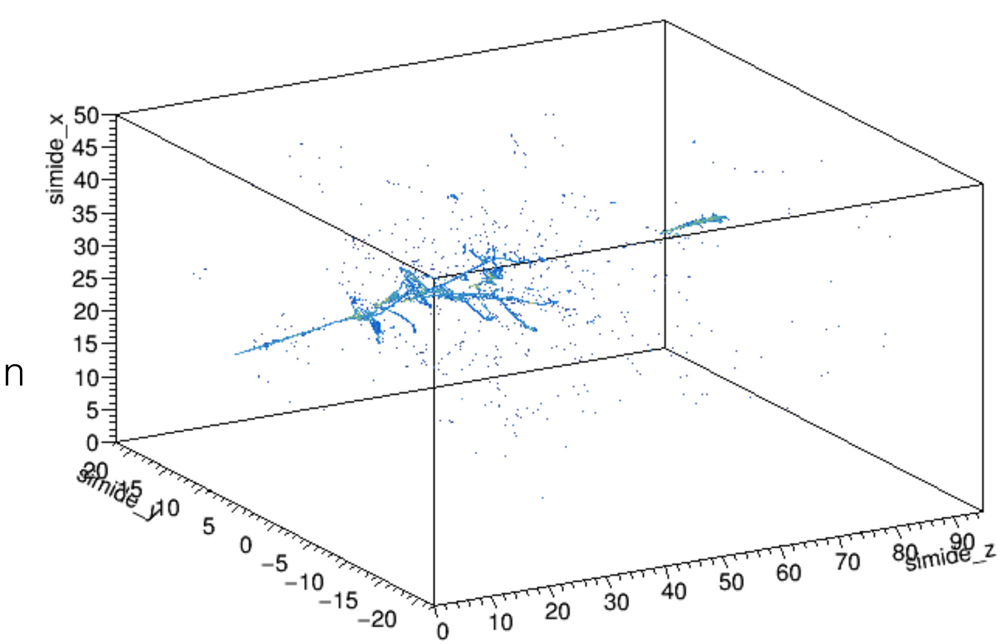


2. Energy Regression (Wednesday)

- Our first attempts didn't give good result.
- Models should estimate error. Account for

3. 2D to 3D (Friday)

- LArTPC wire readout necessary due to heat load.
    - Full Pixelized readout would give ~ N^2 datapoint/time slice
    - Wire readout give ~2N datapoint/time
- Information loss is "recovered" in reconstruction by assuming particle interaction topologies (track, shower, …)
- Tomographic approach (Wirecell) "resolves" ambiguities through costly Markov Chain MC
- Perhaps a DNN can learn the topologies and infer a 3D image

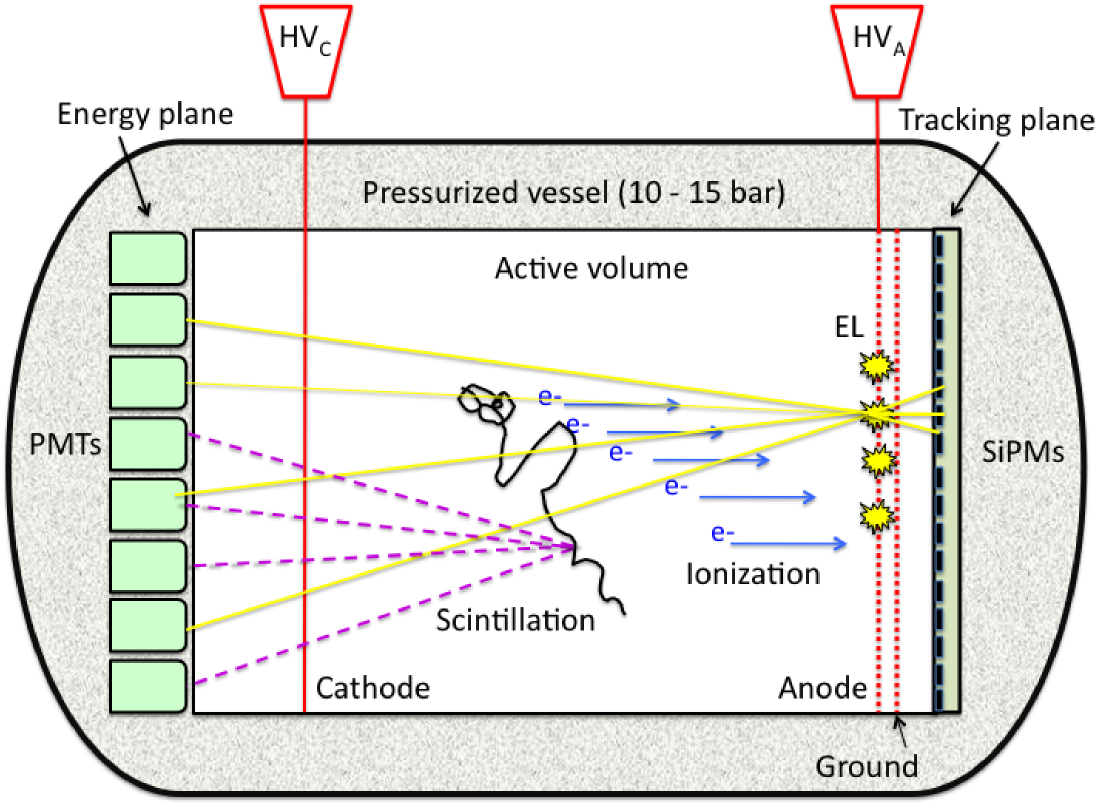


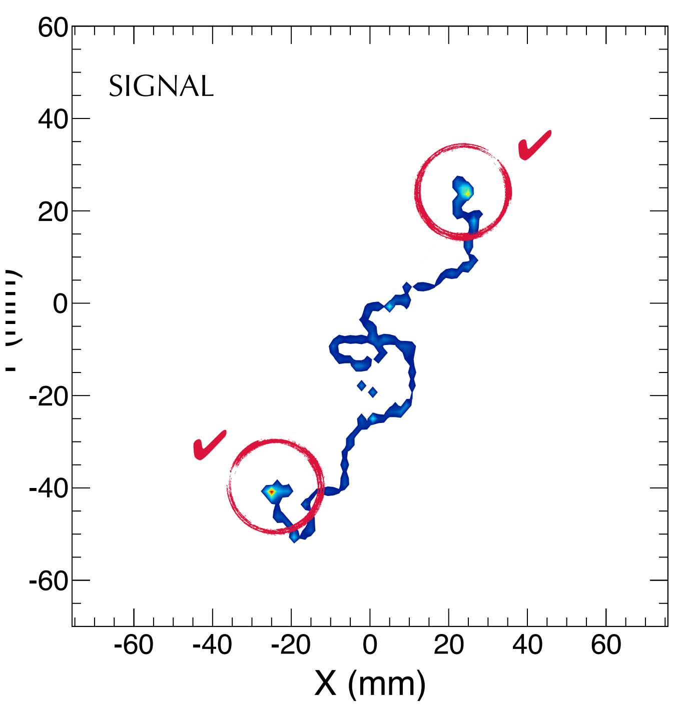
4. Noise suppression…

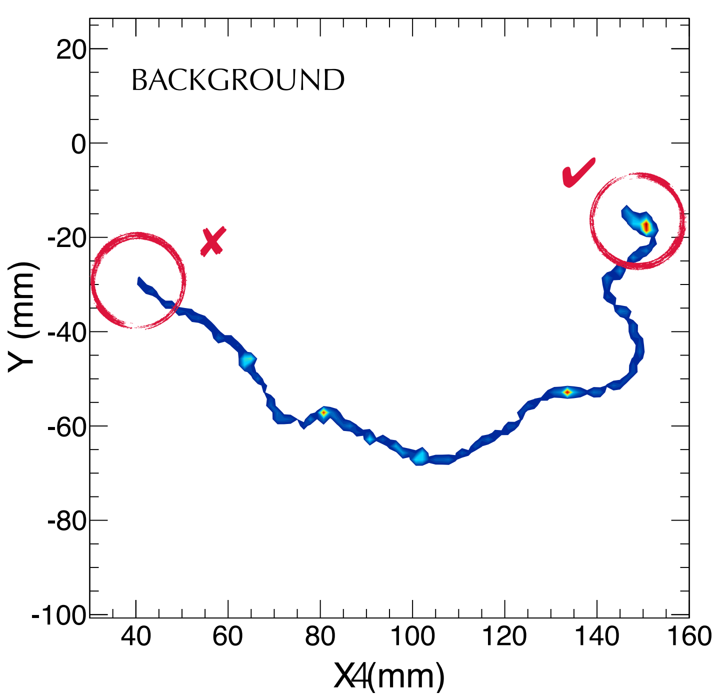
NEXT Experiment

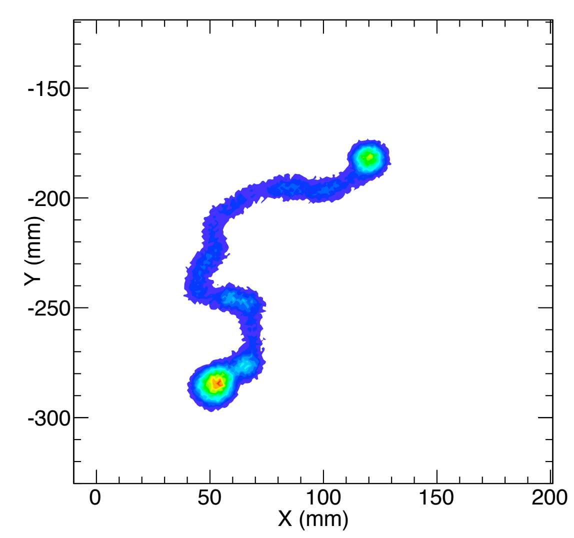
(J. Renner, J.J. Gomez, …, AF)

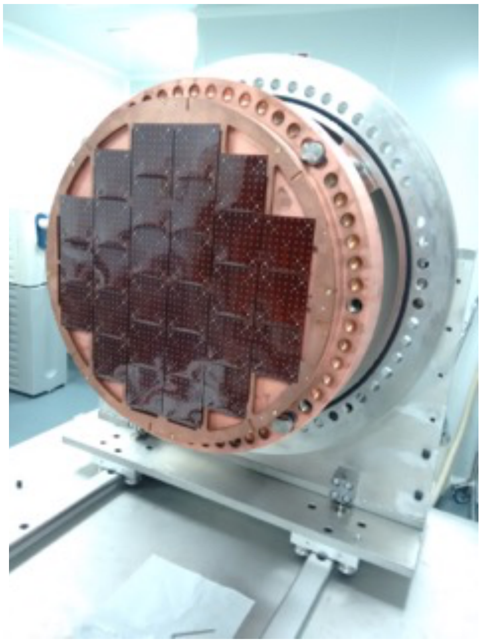
- ***Neutrinoless Double Beta Decay*** using Gas TPC/SiPMs
- Signal: 2 Electrons. Bkg: 1 Electron.
- Hard to distinguish due to ***multiple scattering***.
- ***3D readout***… candidate for 3D Conv Nets.
- Just a handful of signal events will lead to ***noble prize***
    - Can we trust a DNN at this level?







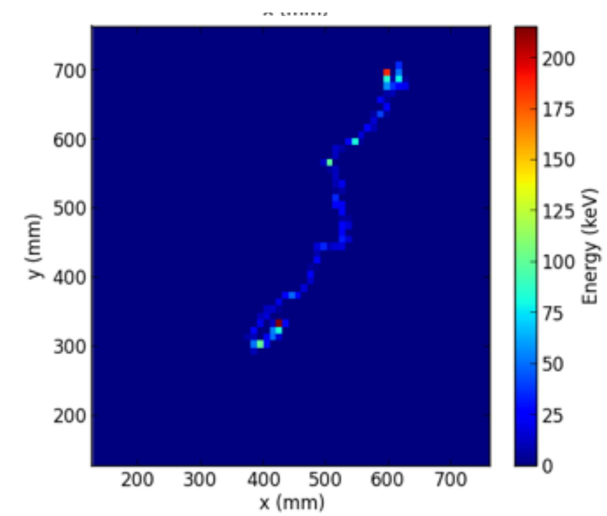


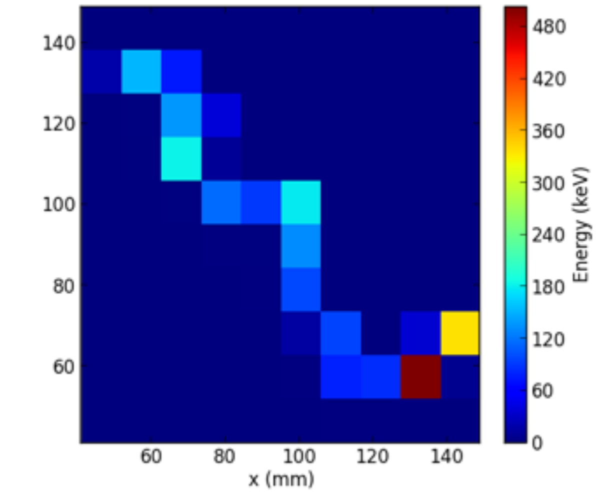


NEXT Detector Optimization

- Idea 1: use DNNs to ***optimize detector***.
  - Simulate data at different resolutions
  - Use DNN to quickly/easily assess best performance for given resolution.

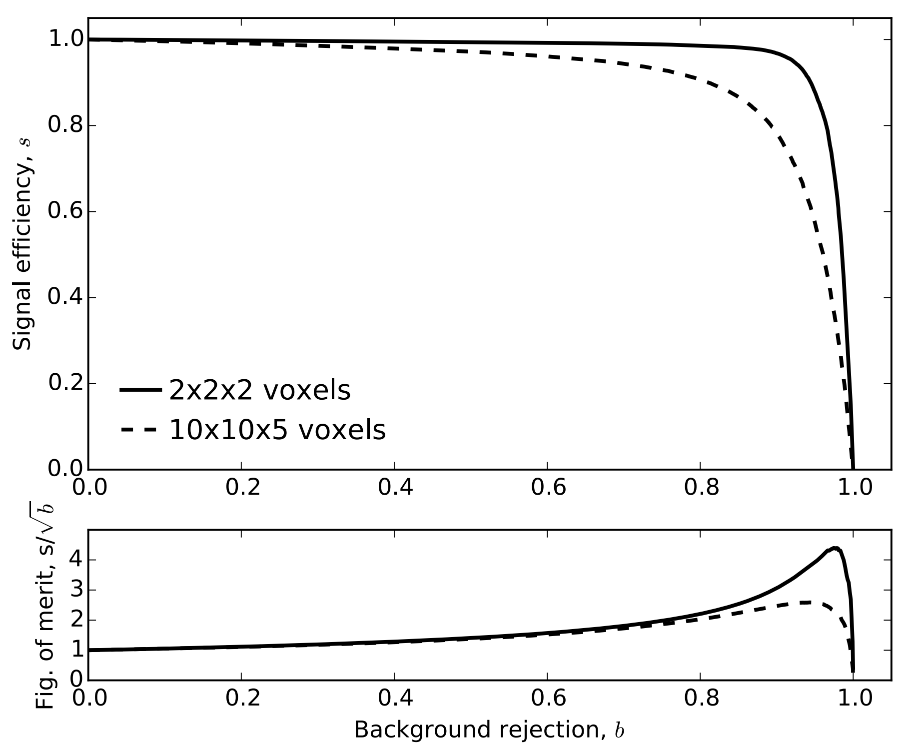
Analysis	Signal eff. (%)	B.G. accepted (%)
DNN analysis (2 x 2 x 2 voxels)	86.2	4.7
Conventional analysis (2 x 2 x 2 voxels)	86.2	7.6
DNN analysis (10 x 10 x 5 voxels)	76.6	9.4
Conventional analysis (10 x 10 x 5 voxels)	76.6	11.0





- Idea 2: ***systematically study*** the relative importance of various physics/detector effects.
  - Start with simplified simulation. Use DNN to assess performance.
  - Turn on effects one-by-one.

2x2x2 voxels	Run description	Avg. accuracy (%)
	Toy MC, ideal	99.8
	Toy MC, realistic $0\nu\beta\beta$ distribution	98.9
	Xe box GEANT4, no secondaries, no E-fluctuations	98.3
	Xe box GEANT4, no secondaries, no E-fluctuations, no brem.	98.3
	Toy MC, realistic $0\nu\beta\beta$ distribution, double multiple scattering	97.8
	Xe box GEANT4, no secondaries	94.6
	Xe box GEANT4, no E-fluctuations	93.0
	Xe box, no brem.	92.4
	Xe box, all physics	92.1
	NEXT-100 GEANT4	91.6
10x10x5 voxels		
	NEXT-100 GEANT4	84.5



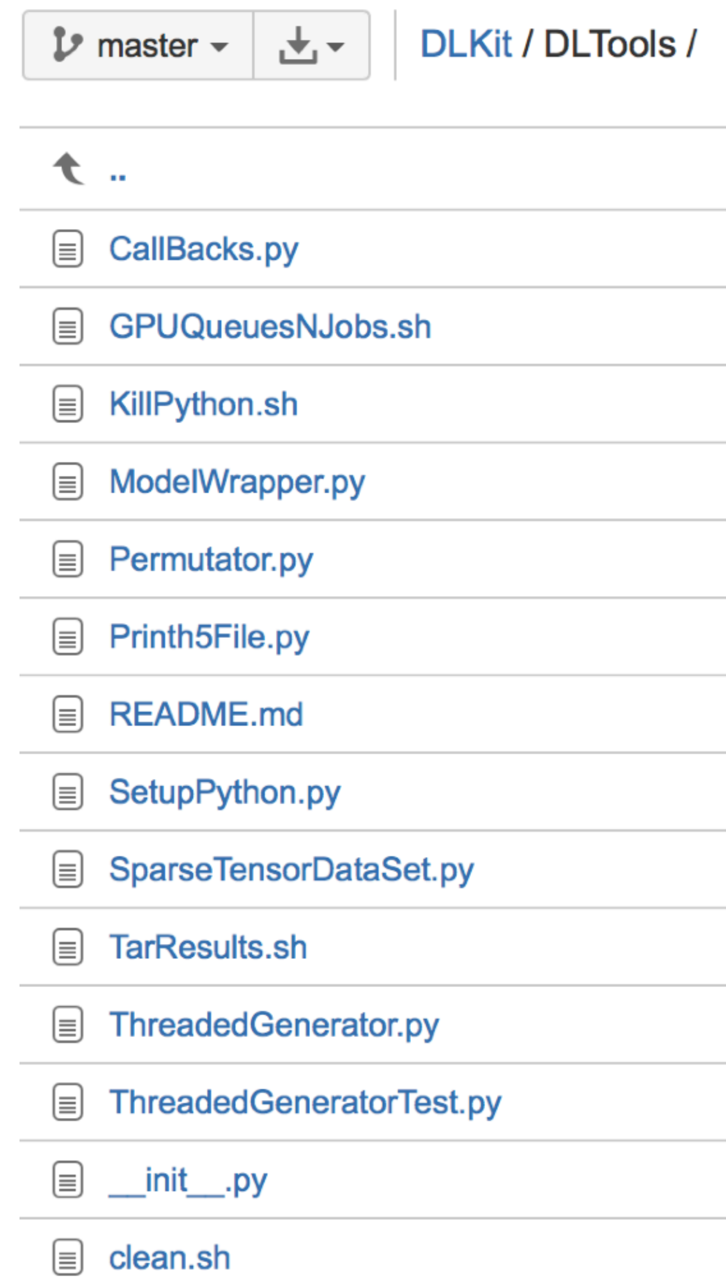
Software

Technical Challenges

- Datasets are too large to fit in memory.
- Data comes as many h5 files, each containing O(1000) events, organized into directories by particle type.
- For training, data needs to be read, mixed, "labeled", possibly augmented, and normalized…. can be time consuming.
- Very difficult to keep the GPU fed with data. GPU utilization often < 10%, rarely > 50%.
- Keras python multi-process generator mechanism has limitations…
- So I wrote a standalone parallel generator… **DLGenerators**:
    - Generic Design:
        - Specify keys of objects you want to read and list of files in each class.
        - Pre-process function: runs in parallel. Good for normalization / reformatting / augmentation
        - Post-process function: not run in parallel. Re-grouping objects to fit network architecture.
    - Simple… useful even when parallelization is not necessary:
        - Handles class/file book-keeping and mixing.
        - Automatically caches data to disk, so 2nd epoch run much faster.
- Scales up to ~40 processes almost linearly…
- Gains for > ~40, but less efficient because file handles collisions.

DLKit

- Thin layer on top of Keras.
- My personal DNN framework. I imagine many of you would write something similar…
- Handles book keeping for comparing large number of training sessions (e.g. for hyper parameter scan or optimization)
  - *Model Wrapper* that book keeps instantiation, training, and evaluation parameters.
  - *Permutator* that produces configurations with unique index.
- Tools necessary to setup HEP problems.
  - *Sparse Tensor*: store sparse N-Dim data or turn particle trajectories into images on fly.
  - *Calls backs*: gracefully stop training based on running time, catching signals, AUC, …
  - *Generators*: for data reading.
  - *Analysis*: standard analysis methods for typical plots.
  - *Loss functions*: for physics regression targets.



CaloDNN / LArTPCDNN / NEXTDNN

- Instantiates generators for efficiently reading or premixing data.
- Provides out-of-the-box running.
- Orchestrates running large HP scans.
  - Makes tables…
  - Jupyter notebook-based analysis.
    - Generates standard plots.
- https://github.com/UTA-HEP-Computing/CaloDNN
- Gearing up for a big BlueWaters run…
  - Large HP Scan (not optimization)
  - “Regularization”: training time.
- Can be configured for other data… let me know if you want to try it with LCD data.

Analysis.py
ClassificationArguments.py
ClassificationExperiment.py
ClassificationScanConfig.py
LCDData.py
Models.py
README.md
ScanJob.py
ScanJob.sh
SubmitMerge.sh
__init__.py
requirements.txt

afarbin — ssh -YX orodruin.uta.edu — 111×29

```
Last login: Tue Feb 28 08:47:35 2017 from 192.168.1.13
afarbin@thecount:~$ cd LCD/DLKit/
afarbin@thecount:~/LCD/DLKit$ source setup.sh
(Keras) afarbin@thecount:~/LCD/DLKit$ python -m CaloDNN.ClassificationExperiment --help
usage: ClassificationExperiment.py [-h] [-C CONFIG] [-L LOADMODEL]
                                   [--gpu GPUID] [--cpu] [--NoTrain]
                                   [--NoAnalysis] [--Test] [-s HYPERPARAMSET]
                                   [--nopremix] [--preload] [-r RUNNINGTIME]

optional arguments:
  -h, --help            show this help message and exit
  -C CONFIG, --config CONFIG
                        Use specified configuration file.
  -L LOADMODEL, --LoadModel LOADMODEL
                        Loads a model from specified directory.
  --gpu GPUID           Use specified GPU.
  --cpu                 Use CPU.
  --NoTrain             Do not run training.
  --NoAnalysis          Do not run analysis.
  --Test                Run in test mode (reduced examples and epochs).
  -s HYPERPARAMSET, --hyperparamset HYPERPARAMSET
                        Use specificed (by index) hyperparameter set.
  --nopremix            Do not use the premixed inputfile. Mix on the fly.
  --preload             Preload the data into memory. Caution: requires lots
                        of memory.
  -r RUNNINGTIME, --runningtime RUNNINGTIME
                        End training after specified number of seconds.
(Keras) afarbin@thecount:~/LCD/DLKit$
```

ScanConfig.py

```
# Input for Premixed Generator
InputFile="/data/afarbin/LCD/LCD-Merged-All.h5"
# Input for Mixing Generator
FileSearch="/data/afarbin/LCD/*/*.h5"

# Generation Model
Config={
    "GenerationModel":"'Load'",
    "MaxEvents":int(3.e6),
    "NTestSamples":100000,
    "NClasses":4,

    "Epochs":1000,
    "BatchSize":1024,

    # Configures the parallel data generator that read the input.
    # These have been optimized by hand. Your system may have
    # more optimal configuration.
    "n_threads":4,  # Number of workers
    "multiplier":2, # Read N batches worth of data in each worker

    # How weights are initialized
    "WeightInitialization":"'normal'",

    # Normalization determined by hand.
    "ECAL":True,
    "ECALNorm":150.,

    # Normalization needs to be determined by hand.
    "HCAL":True,
    "HCALNorm":150.,
```

```
    # Set the ECAL/HCAL Width/Depth for the Dense model.
    # Note that ECAL/HCAL Width/Depth are changed to "Width" and "Depth",
    # if these parameters are set.
    "HCALWidth":32,
    "HCALDepth":2,
    "ECALWidth":32,
    "ECALDepth":2,

    # No specific reason to pick these. Needs study.
    # Note that the optimizer name should be the class name (https://keras.io/optimizers/)
    "loss":"'categorical_crossentropy'",

    # Specify the optimizer class name as True (see: https://keras.io/optimizers/)
    # and parameters (using constructor keywords as parameter name).
    # Note if parameter is not specified, default values are used.
    "optimizer":"'SGD'",
    #"lr":0.01,
    #"decay":0.001,

    # Parameter monitored by Callbacks
    "monitor":"'val_loss'",

    # Active Callbacks
    # Specify the CallBack class name as True (see: https://keras.io/callbacks/)
    # and parameters (using constructor keywords as parameter name,
    # with classname added).
    "ModelCheckpoint":True,
    "Model_Chekpoint_save_best_only":False,

    # Configure Running time callback
    # Set RunningTime to a value to stop training after N seconds.
    "RunningTime": 3600,
}
```

```

# Parameters to scan and their scan points.
Params={ "Width":[32,64,128,256,512],
         "Depth":range(1,5),
         "lr":[0.1,0.01,0.001],
         "decay":[0.1,0.01,0.001],
          }

```

```
(Keras) afarbin@thecount:~/LCD/DLKit$
(Keras) afarbin@thecount:~/LCD/DLKit$
(Keras) afarbin@thecount:~/LCD/DLKit$ python -m DLTools.ScanAnalysis TrainedModels.TestScan.1/
Using Theano backend.
                         Ele_AUC    Width    Depth    Pi0_AUC    ChPi_AUC    Gamma_AUC
----------------------  ---------  -------  -------  ---------  ----------  -----------
CaloDNN_32_1_Merged.23     0.9452       32        1     0.8608      0.9971       0.8802
CaloDNN_128_1_Merged.1     0.9639      128        1     0.9151      0.9964       0.9299
CaloDNN_64_1_Merged.1      0.9810       64        1     0.9453      0.9975       0.9508
CaloDNN_256_1_Merged.1     0.9870      256        1     0.9529      0.9987       0.9494
(Keras) afarbin@thecount:~/LCD/DLKit$
```

jupyter AnalyzeScan (autosaved)

Logout

File Edit View Insert Cell Kernel Help

Python 2

CellToolbar

```
acc', 'All_History.loss', 'All_History.val_loss', 'All_History.acc', 'All_Depth']
```

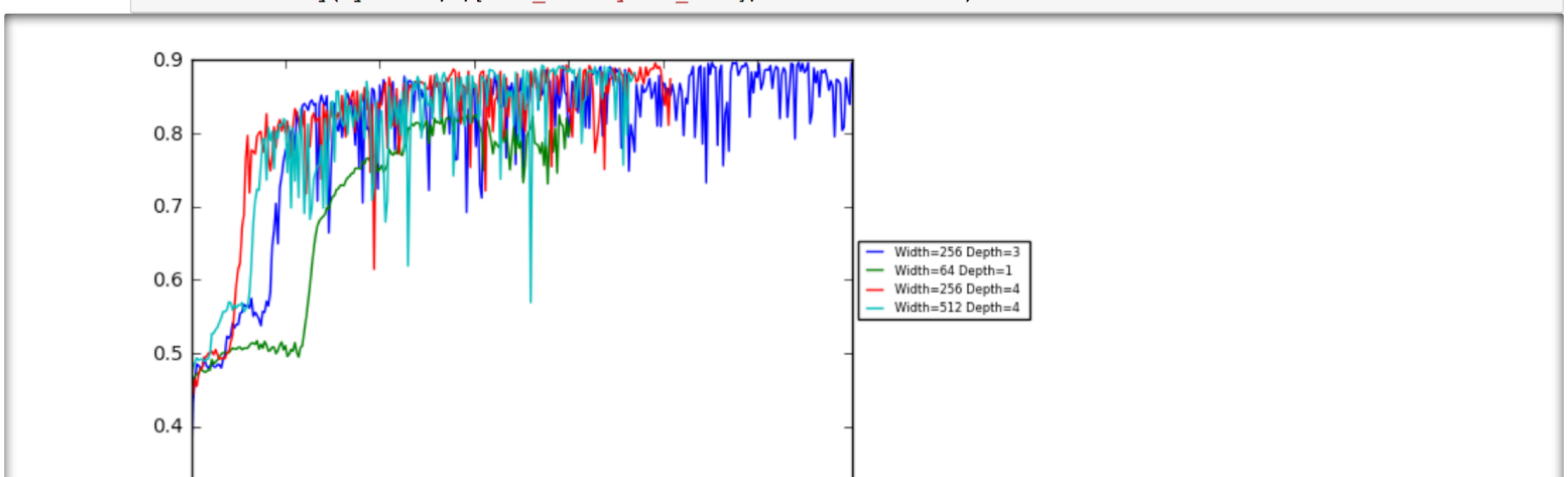
In [4]:
```python
# Get a List of all numbers stored in MetaData
print "Available Parameters:", GetGoodParams(MyModels)
```

```
Available Parameters: ['Ele_AUC', 'Width', 'Depth', 'Pi0_AUC', 'Epochs', 'Gamma_AUC', 'ChPi_AUC']
```

In [5]:
```python
# Make a Table of all relevant parameters, sort by 1,2,then 0 columns.
# Note: Parameters are optional... but the columns and rows will be not optimally sorted.
ScanTable(MyModels,['Model Name', 'Width', 'Depth', 'Epochs', 'Ele_AUC', 'Pi0_AUC', 'ChPi_AUC', 'Gamma_AUC'],[1,2,0])
```

Model Name	Width	Depth	Epochs	Ele_AUC	Pi0_AUC	ChPi_AUC	Gamma_AUC
Width=32 Depth=1	32	1	228	0.9183	0.8657	0.9916	0.8833
Width=32 Depth=2	32	2	335	0.9364	0.8382	0.9885	0.8583
Width=32 Depth=3	32	3	298	0.9404	0.8979	0.9864	0.8572
Width=32 Depth=4	32	4	320	0.9139	0.8879	0.9518	0.8639
Width=64 Depth=1	64	1	251	0.9262	0.8712	0.9961	0.9022
Width=64 Depth=2	64	2	304	0.9320	0.9015	0.9887	0.9078
Width=64 Depth=3	64	3	432	0.9388	0.9164	0.9922	0.8186
Width=64 Depth=4	64	4	339	0.9808	0.9372	0.9983	0.9414
Width=128 Depth=1	128	1	342	0.9715	0.9154	0.9966	0.9357
Width=128 Depth=2	128	2	213	0.9500	0.8650	0.9956	0.9083
Width=128 Depth=3	128	3	318	0.9627	0.9322	0.9934	0.9261
Width=128 Depth=4	128	4	450	0.9879	0.9198	0.9984	0.9335
Width=256 Depth=1	256	1	395	0.9783	0.9191	0.9978	0.9436
Width=256 Depth=2	256	2	365	0.9473	0.9199	0.9913	0.9103
Width=256 Depth=3	256	3	437	0.9798	0.9544	0.9969	0.9570
Width=256 Depth=4	256	4	294	0.9276	0.9025	0.9859	0.9034
Width=512 Depth=1	512	1	292	0.9397	0.8777	0.9858	0.9067
Width=512 Depth=2	512	2	325	0.9588	0.9355	0.9868	0.9224
Width=512 Depth=3	512	3	289	0.9762	0.9339	0.9972	0.9195
Width=512 Depth=4	512	4	308	0.9349	0.8710	0.9921	0.8861

In [6]:
```python
# Plot Historical MetaData... put 4 models per plot
#PlotMetaDataMany(MyModels,4,["History","val_loss"],loc="center left")
PlotMetaDataMany(MyModels,4,["All_History.val_acc"],loc="center left")
```



In [7]:
```python
# Compare Number of Epochs each model ran (only last run)
PlotMetaData(MyModels,["Epochs"])
```

jupyter AnalyzePerformance (autosaved)

Logout

File Edit View Insert Cell Kernel Help

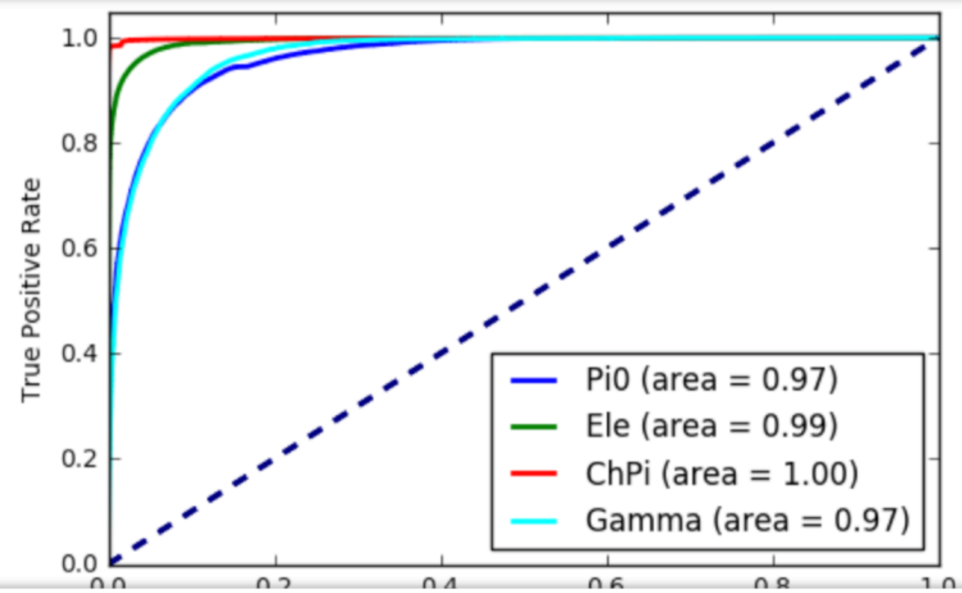
No kernel

Python 2

Code

CellToolbar

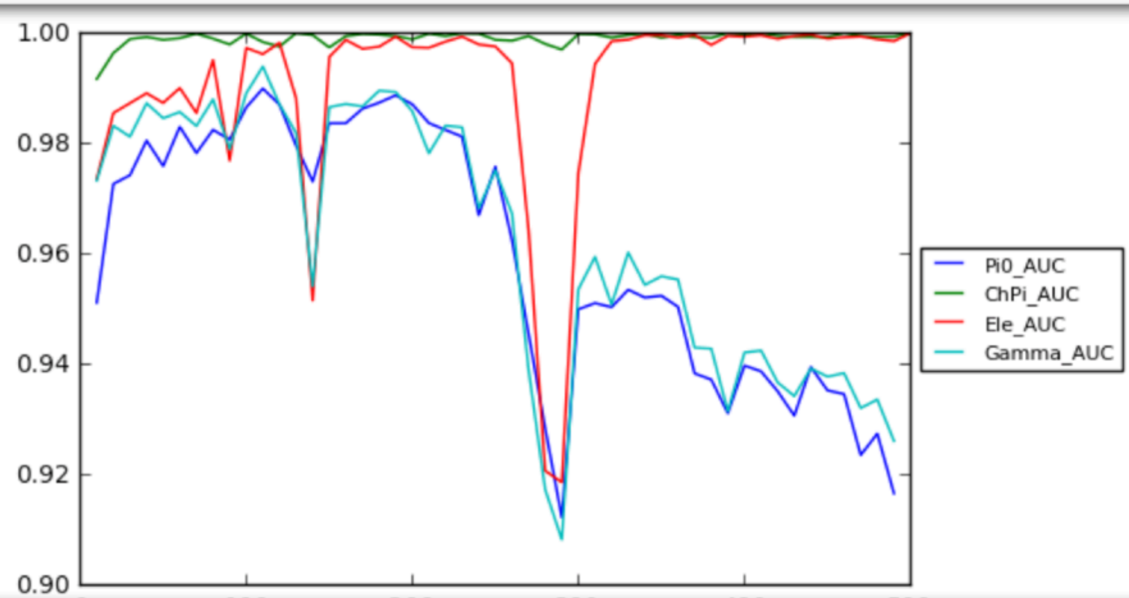
```
# Also performs inference on the test data, returning the results
from DLAnalysis.Classification import *
result,NewMetaData=MultiClassificationAnalysis(MyModel,[Test_X_ECAL,Test_X_HCAL],Test_Y,BatchSize,
                                               IndexMap={0:'Pi0', 2:'ChPi', 3:'Gamma', 1:'Ele'})
```



In [4]:
```
# Bin the data
Energy=target[:,:,2].flatten()

def AUCvsEnergy(E_min=10.,E_max=510.,E_bins=100.):
    BD,E_binning=BinDataIndex(Energy, E_min, E_max, E_bins)
    # Run the Classification Analysis in Bins
    return BinMultiClassificationAnalysis(MyModel,Test_Y=Test_Y,Y_binning=E_binning,
                                          bin_indecies=BD, result=result,
                                          IndexMap={0:'Pi0', 2:'ChPi', 3:'Gamma', 1:'Ele'})
```

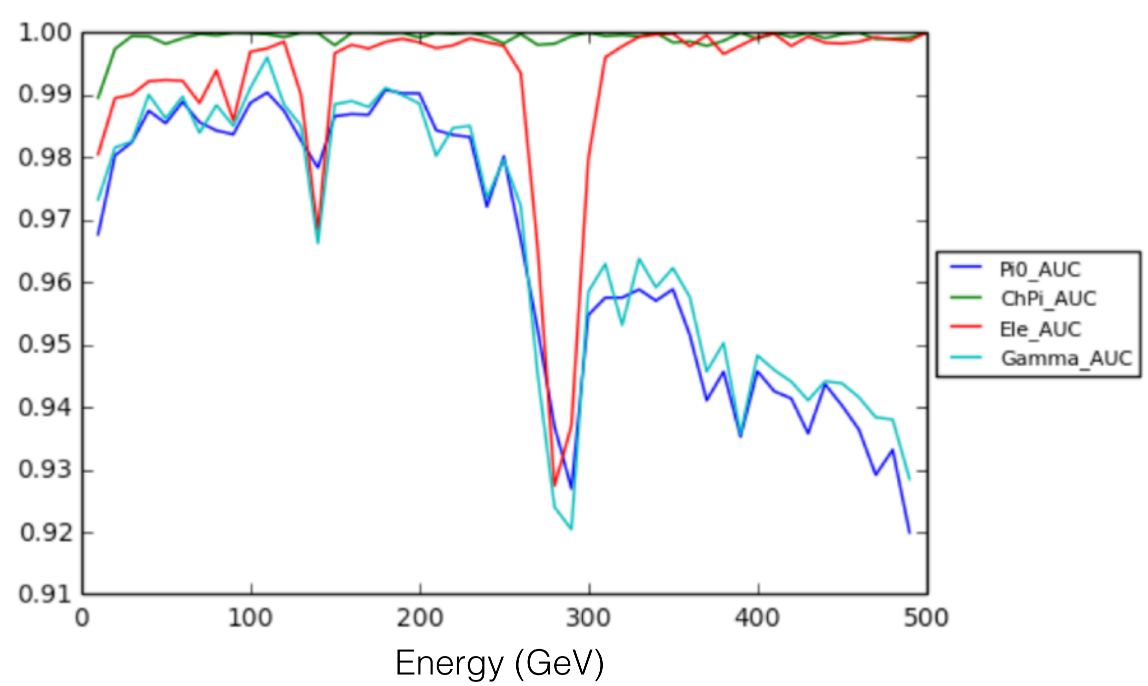
In [5]:
```
# Full Energy Range
Res=AUCvsEnergy(10.,510.,50.)
```



In [6]:
```
# 10 to 100 GeV
```

Example Results

Model Name	Width	Depth	Epochs	Ele_AUC	Pi0_AUC	ChPi_AUC	Gamma_AUC
Width=32 Depth=1	32	1	27	0.9857	0.9560	0.9977	0.9569
Width=32 Depth=2	32	2	16	0.9843	0.9502	0.9986	0.9542
Width=32 Depth=3	32	3	12	0.8092	0.8077	0.9972	0.7605
Width=32 Depth=4	32	4	16	0.7009	0.7617	0.9973	0.6510
Width=64 Depth=1	64	1	26	0.9875	0.9567	0.9985	0.9616
Width=64 Depth=2	64	2	15	0.9887	0.9571	0.9988	0.9586
Width=64 Depth=3	64	3	24	0.9865	0.9564	0.9986	0.9602
Width=64 Depth=4	64	4	31	0.9874	0.9584	0.9986	0.9593
Width=128 Depth=1	128	1	26	0.9923	0.9672	0.9991	0.9661
Width=128 Depth=2	128	2	16	0.9934	0.9687	0.9992	0.9695
Width=128 Depth=3	128	3	38	0.9938	0.9691	0.9992	0.9701
Width=128 Depth=4	128	4	12	0.9922	0.9643	0.9990	0.9652
Width=256 Depth=1	256	1	24	0.9929	0.9696	0.9991	0.9685
Width=256 Depth=2	256	2	19	0.9945	0.9711	0.9991	0.9707
Width=256 Depth=3	256	3	11	0.9945	0.9674	0.9992	0.9678
Width=256 Depth=4	256	4	33	0.9947	0.9691	0.9992	0.9696
Width=512 Depth=2	512	2	29	0.9951	0.9711	0.9991	0.9715
Width=512 Depth=3	512	3	23	0.9954	0.9690	0.9993	0.9696
Width=512 Depth=4	512	4	16	0.9943	0.9661	0.9990	0.9666



UTA-DL Cluster

- Register for accounts:
  - https://www.utadl.org
  - Once we approve, you'll get an email.
- Machines
  - Oscar: (head node)
    - 6-core Xeon
    - 2 GPUs (Kepler/Maxwell)
  - Thingone/Thingtwo:
    - 6-core i7
    - 4 GTX 1080s in each
  - Super:
    - 2x 12-core Xeon
    - 4 GTX 1080s
  - TheCount:
    - 2x 22-core xeon
    - 10Titan X (Pascal)
- 100 TB storage. 10G network. SSD cache on every machine.

- Request account:
  - https://www.utadl.org
  - wait for email.
- Create tunnel:
  - ssh -NfL 8000:localhost:8000 <username>@orodruin.uta.edu
- Point browser to: 127.0.0.1:8000